ASCII SYSTEMSOFT

PCファミリー・テクニカル・ノウハウ集 PC-8800シリーズ編第1巻

PC-TechKnow8800 Vol.1

栗山・平松・松尾共著 システムソフト監修

アスキー出版局

アスキー・システムソフトシリーズ
PCファミリー・テクニカル・ノウハウ集

PC-Techknow8800Vol.1

共著／栗山浩一　平松達雄
　　　松尾篤弥
監修／システム　ソフト

アスキー・システムソフトシリーズ

PCファミリー・テクニカル・ノウハウ集の発刊にあたって

株式会社システムソフト福岡では、株式会社アスキー出版と共同で、「アスキー・システムソフトシリーズ、PCファミリー・テクニカル・ノウハウ集」を発刊することになりました。

この、「PC-Tech Knowシリーズ」は、NECのパーソナルコンピュータ、PCファミリー（PC-8001、PC-6001、PC-8801）および、その周辺機器を徹底的に活用するためのテクニカル・ノウハウ（Tech Knowテクノウ）をまとめたもので、構成は以下のようになっています。

○PC-8000シリーズ編（N-BASIC）
○PC-6000シリーズ編（N60-BASIC）
○PC-8800シリーズ編（N88-BASIC）

各シリーズ共、2巻程度にまとめる予定です。

内容は、それぞれのBASICの内部構造から、キー入力の仕方、カセットおよびディスクファイルの上手な使用法、BASICプログラム・テクニック、さらには、機械語理解のポイントまで、活用いただける情報が満載されています。

本書は、PC-8800シリーズ第1巻の「PC-Techknow8800 Vol.1」であり、2巻以後も順次発刊の予定です。

はじめに

日本電気から最初のパソコンＰＣ-8001が発表されたのが、1979年５月に開かれた第１回マイコンショーのことです。

その後、ホビーユースを対象とした低価格パソコンＰＣ-6001、ビジネス機能をアップした本格的パソコンＰＣ-8801が加わり、選べる３機種、３機能を銘打ってＰＣ-ファミリーができあがりました。

この中でも、ＰＣ-8801は、究極の８ビットマシンと言われるくらい並すぐれたハードウェア、ソフトウェアを有し、高度なグラフィック処理、日本語表示を可能にしています。

これだけの材料が与えられると、後は料理人（あなたですよ。）がいかにこの材料をうまく使いこなして、おいしい料理を作るかということですが、材料の豊富さゆえにその調理法も複雑なものとなっています。(あの３冊のぶ厚いマニュアルを見てげんなりした人も多いのではありませんか？)

おいしい料理を作るには、まず材料を知ることからというのは、パソコンのプログラミングにおいても同じことです。

そこで本書では、料理法（プログラミングの仕方）というよりも、材料の説明ということで、ＰＣ-8801本体はもちろん、プリンタ、ディスクユニットに至るまでＰＣ-8800シリーズの機能を徹底解剖し、内部構造、より高度な活用のためのノウハウ、各種資料をまとめました。

基本的なことにもなるべく触れるように心掛けたつもりですが、限られた紙面の中ですべてを書き尽くすことは不可能であり、難解な点があることを御容赦下さい。

なお、本書の原稿作成は、栗山、平松が共同で行い、また松尾は、ハードウェアの立場から執筆に参加しました。

最後になりましたが、本書を出版するにあたり、株式会社アスキー出版の編集スタッフの方々、株式会社システムソフト福岡の樺島社長、藤田出版部長には、大変なお世話になりました。この場を借りて心から御礼申し上げます。

1982年12月

※本文および付録に記載されている内容については、筆者らが独自に調査、解析したものであり、運用上の影響については責任を負いかねますので御了承ください。

目　次

目　次

目　次

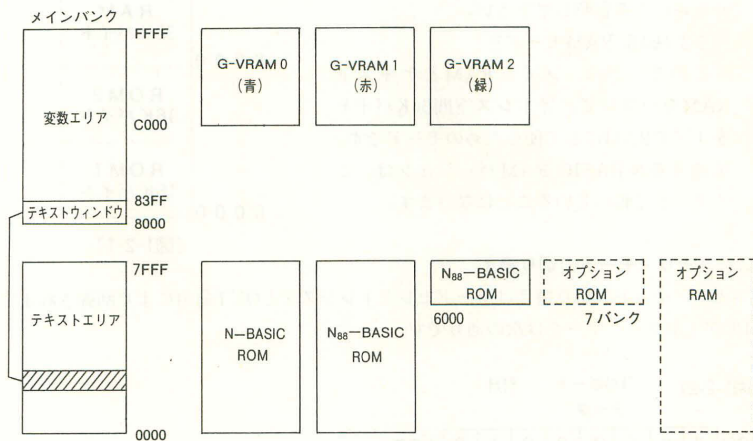
第1章　メモリマップとテキスト・ウィンドウ

第1章　メモリ・マップとテキストウインドウ

1-1. メモリ・マップ

PC-8801は、標準でRAM112Kバイト、ROM72Kバイト、の合計184Kバイトにも及ぶメモリを持っています。

これらは、64KバイトしかないCPUのアドレス空間上に図1-1-1のように置かれ、バンク切り換えによって制御されます。

（図1-1-1）

1-2. メモリ・モード

1-2-1 3つのメモリ・モード

PC-8801では、次の3つのメモリ・モードがあります。

(1)モード0　　N-BASICモード

(2)モード1　　N_{88}-BASICモード

(3)モード2　　64K RAMモード

モード0とモード1は、本体後部のDIPスイッチにより選択されます。

(1)モード0（N-BASICモード）

このモードは、PC-8001のN-BASICと同じモードです。図1-2-1のようなメモリ・マップとなり、PC-8001上で動くプログラムは、このモードで使用できます。

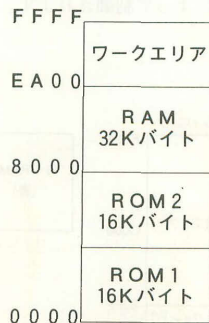
ただし、PC-8001では、6000H番地から、7FFFH番地までは、拡張ROMエリアとして使うことができましたが、PC-8801では、モニタプログラム（N-BASICからは使えません）や、N-BASICの追加部分が置かれているため使用できません。

(2)モード1（N_{88}-BASICモード）

このモードでは、N_{88}-BASICが動きます。

このときのメモリ・マップなどについては、第2章で詳しく解説してありますので、そちらの方を参照して下さい。

(3)モード2（64K RAMモード）

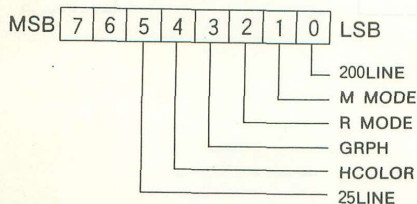
このモードは、メインRAMとテキストRAMをつなげて、アドレス空間64KバイトをすべてRAMにして使うためのモードです。後述するN-BASIC RAMバージョンは、このモードで動いていることになります。

(図1-2-1)

1-2-2 メモリ・モードの切り換え

メモリ・モードの切り換えはモードセレクトレジスタとOUT命令により制御されます。I/Oアドレスと、データは次の通りです。

(図1-2-2)　　I/0ポート　31H
データ

MSB 7 6 5 4 3 2 1 0 LSB

0 — 200LINE
1 — M MODE
2 — R MODE
3 — GRPH
4 — HCOLOR
5 — 25LINE

	R MODE	M MODE
モード0	1	0
モード1	0	0
モード2	—	1

(図1-2-2)

メモリ・モードの制御に必要なのは、bit2とbit1だけですが、他のbitを勝手に変えると画面が乱れたりすることがありますので注意して下さい。

N88-BASICでは、前にI/Oポート31Hに出力したデータの値をワークエリア上（E6C2H番地）に持っていますので、64KRAMモード、N88-BASICモード間の切り換えを行なうには、次のようにすると簡単です。

○N88-BASICモード⇨64K RAMモード

```
DI
LD      A,(0E6C2H)
OR      6
OUT     (31H),A
LD      (0E6C2H),A
EI
```

○64K RAMモード⇨N88-BASICモード

```
DI
LD      A,(0E6C2H)
AND     0F9H
OUT     (31H),A
LD      (0E6C2H),A
EI
```

1-2-3 N-BASICモードからN88-BASICモードへ

N88-BASICモードからN-BASICを起動するには、NEW ON 1を実行すればよいわけですが、その逆の命令はありません。

後ろのDIPスイッチが、N88-BASICモードであれば、リセットですむのですが、そうでない場合は困ります。

そこで、N-BASICモードからN88-BASICを起動する方法を紹介しましょう。

モニターモードで次のプログラムを実行するとN88-BASICモードに移ります。

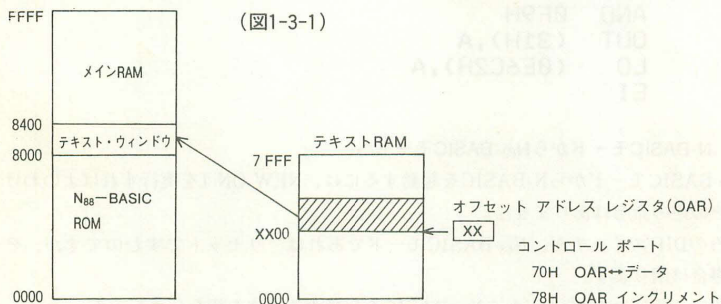
```
C000   AF              XOR  A
C001   D3 31           OUT  (31H),A
C003   3E F3           LD   A,0F3H
C006   C3 FD 77        JP   77FDH
```

このプログラムはどこにでもおけますので、両方のBASICで使用されていないエリア（例えば、F2F0H番地から）などに書き込んでおくとよいでしょう。

1-3. テキスト・ウィンドウの使い方

N88-BASICではテキストエリアが0000～7FFFHというBASIC ROMと同じアドレスに割り当てられています。ということはBASICインタプリタがテキストを見ようとしても見ることができません。これを解決するために「テキスト・ウィンドウ」というものを導入しています。これはテキストの一部分を8000～83FFHのアドレスに投影し、あたかも窓からテキストエリアを見る様な状態にします。つまり、テキストエリアのある所を見たい場合には、その場所をウィンドウに映し出して見るわけです。なお、このウィンドウはWINDOW文のウィンドウともCONSOLE文で制御するウィンドウともまったく別のものです。

図1-3-1を見て下さい。まず、オフセットアドレス・レジスタ（OAR）というものがあります。ここに8ビットのデータを入れると、そのデータを上位8ビットとしたアドレス（下位8ビットは00）から1Kバイトがウィンドウに現れます。たとえばOARに1FHを入れると1F00～22FFHがウィンドウに現れます。35Hを入れると3500～38FFHです。このOARはI/Oポートの70H番地に割り当てられています。70H番地にデータをOUTすると、OARにその値がセットされます。

（図1-3-1）

では実際にやってみましょう。まず次のプログラムを実行して下さい。

```
10 FOR I=&H4000 TO &H40FF
20   POKE I,I MOD 256
30 NEXT
```

何かROMエリアにPOKEしているようですが、実際にはこのデータはテキストRAMに書き込まれます。テキストRAMは拡張RAMを使わない限り、いつも書き込みができる状態にあるからです。言うまでもなくROMには書き込まれません。次にI/Oポート70Hに40Hを出力すれば4000～43FFHが8000～83FFHに現れるのですが、BASICのOUT文ではうまくいかないことがあります。確かにOARには書き込まれるのですが、そのあとOKを表示する時にN88-BASICインタプリタがOARをリセットしてしまう場合があるからです。そこでモニターに入ります。

mon⏎

h］o70，40⏎

これでOARに40Hが入りました。

h］e8000⏎

```
8000 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
8010 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
8020 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
8030 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
8040 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
8050 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
8060 60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
8070 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
8080 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
8090 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
80A0 A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
80B0 B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
80C0 C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
80D0 D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
80E0 E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF
80F0 F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF
```

ごらんの通り、4000H番地が8000H番地に映し出されました。

今、エディットモードになっていますが、ここでメモリを書き換えたらどうなるでしょうか。8000H番地からのメインRAMに書き込まれるのでしょうか？いいえ、これもちゃんと4000H番地からに書き込まれるのです。

では、テキスト・ウィンドウを使っている時は本来の8000～83FFHのRAMはどうなるかといいますと、CPUのアドレス空間からまったく切り離され、読むことも書くこともできません。でも何とかして読めないでしょうか…そう、これもウィンドウに映せばよいのです。

```
[ESC]
h］o70，80⏎
h］e8000⏎
```

今、映っているのが、8000H番地からのRAM上のデータです。場合によってはN-BASICのプログラムが残っていることもあるでしょう。

OARを制御するI/Oポートとしては70Hの他に78Hがあります。ここに何かデータを出力するとOARの値が１つ増えます。データの値は関係ありません。またOARの値を読むときには、70Hから読めます。

〈例〉

```
mon⏎
h］o70，33⏎
h］o78，0⏎
h］i70⏎
34
h］
```

なお、テキスト・ウィンドウは、ハード的にはN_{88}-BASICモードの時にのみ使用できます。

1-4. 拡張ROMと拡張RAM

1-4-1 拡張ROM

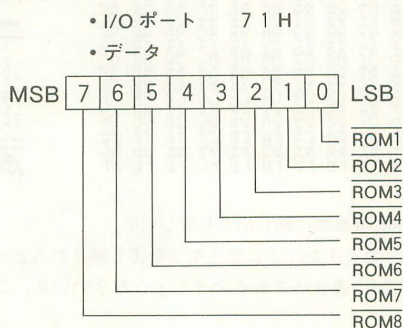
拡張ROMエリアとしては、6000H番地から7FFFH番地までの8Kバイトが割り当てられ、このエリア上に最大8バンクのROMを持つことができます（ただし、このうち1つは、N_{88}-BASICインタプリタの一部として使われているため、実際には、7バンク56Kバイトが最大）。

拡張ROMは、I/Oポート71Hの各ビットで操作します。

（図1-4-A）　　　• I/O ポート　　71 H

• データ

MSB | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | LSB

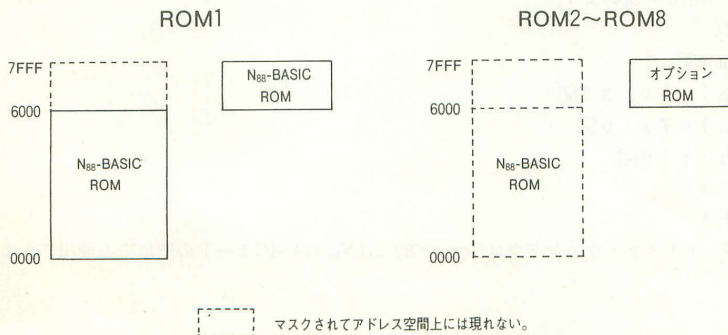
- 0 → $\overline{ROM1}$
- 1 → $\overline{ROM2}$
- 2 → $\overline{ROM3}$
- 3 → $\overline{ROM4}$
- 4 → $\overline{ROM5}$
- 5 → $\overline{ROM6}$
- 6 → $\overline{ROM7}$
- 7 → $\overline{ROM8}$

0 ：指定したROMがセレクトされる
1 ：セレクトされない

例えば、ROM1を選択するには、次のようにします。

```
3E FE    LD   A, 0FEH
D3 71    OUT  (71H), A
```

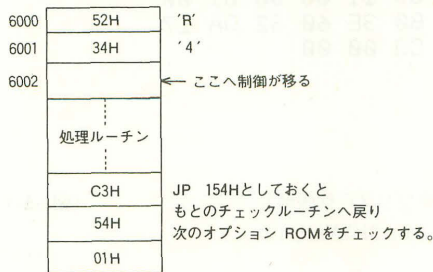
また、ROMの選択状況は、I/Oポート71Hに対して、IN命令を実行することで知ることができます。

拡張ROMがセレクトされると、$\overline{\text{ROM KILL}}$信号により、メインROMはすべてマスクされてしまいますが、ROM1の場合だけは、0H番地から5FFFH番地がアドレス空間上に現われるようになっています。

（図1-4-B）

• 拡張ROMとイニシャライズ

N_{88}-BASICでは、リセット時のイニシャライズで、拡張ROMをチェックしています。

このチェックルーチンでは、拡張ROMの先頭、すなわち6000H番地に52H、次の6001H番地に34Hが書き込まれていると、そのROMの6002H番地に制御を移すようになっています。このチェックは、ROM2からROM8まですべてのバンクについて行なわれます。

(図1-4-C)

1-4-2 拡張RAM

PC-8801では、PC-8001と同じように、拡張RAMを持つことができます。アドレスは、0H番地から、7FFFH番地の32Kバイトです。

拡張RAMとしては、PC-8012-02が使えますが、PC-8801で使用するときは制限があります。

PC-8012-02は、PC-8001で使う場合、読み込み、書き込みを別々に制御できますが、PC-8801では、テキストRAMがあるために、書き込みだけを許すモードは使えません。つまりN_{88}-BASICのPOKE文で、データを書き込むことができないわけです。

したがって、PC-8012-02を使う場合は、書き込み及び読み出しを許可するモードにする必要があります。

• 切り換え方法

[例] • PC-80i2-02をバンク0として使う

```
LD    A, 11H
OUT (0E2H), A
```

• N_{88}-BASICに戻す

```
XOR  A
OUT (0E2H), A
```

なお、PC-8012-02をアクティブにしてデータを書き込んでも、テキストRAMに書き込まれることはありません。

1-5. N-BASICモードでフリーエリアを増やす

N-BASICモードでは、テキストRAMは使われていません。そこで、この余っているRAMを使ってN-BASICモードでのフリーエリアを増やす方法を考えてみましょう。

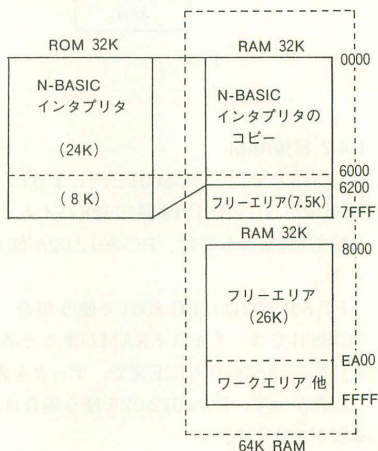
PC-8001の場合は次のプログラムで簡単に行えましたが、PC8801では、6000H番地以降にもN-BASICインタプリタの一部が置かれているため、このままでは使えません。

（PC-8001＋PC-8011の場合）

```
C000 21 00 00 11 00 00 01 00
C008 60 ED B0 3E 60 32 DA 17
C010 D3 E2 C3 00 00

*GC000
```

PC-8801のN-BASICインタプリタの拡張部分は、約280バイトです（N_{88}-BASIC Ver1.1の場合）。

そこで、図1-5-1にあるように、この拡張部分を、6000H番地から、61FFH番地の間へリロケートすることにより、6200H番地から、6FFFH番地までの7.5Kバイトをフリーエリアとして使えるようにします。

（図1-5-1）

次に示すプログラムはこの処理を行なうものです。

```
100 '
110 '---- PC-8801 [ N-BASIC MODE +7.5K ] ----
120 '
130 RESTORE 190
140 FOR I=&HFF50 TO &HFF62
150   READ DA$ : POKE I,VAL("&H"+DA$)
160 NEXT I
170 DEF USR=&HFF50 : DM=USR(0)
180 '
190 DATA 21,00,00,11,00,00,01,00,60,ED,B0,C9,3E,02,D3,31
200 DATA C3,00,00
210 '
220 RESTORE 270
230 FOR I=&H6000 TO &H611F
240   READ DA$ : POKE I,VAL("&H"+DA$)
250 NEXT I
260 '
270 DATA 3E,0B,CD,7C,01,3E,07,CD,83,01,3E,EF,CD,83,01,AF
280 DATA CD,83,01,3E,01,CD,83,01,CD,E9,01,2F,E6,F0,FE,10
290 DATA C9,3E,0F,18,01,AF,F5,3A,C9,ED,FE,FB,28,0F,CD,00
300 DATA 60,20,0A,3E,17,CD,7C,01,F1,CD,83,01,F5,F1,C9,CD
310 DATA 25,60,11,00,C0,C9,3A,C7,ED,B7,28,11,3E,91,CD,29
320 DATA 02,3E,04,32,CB,ED,AF,CD,7C,01,CD,21,60,3A,55,EB
330 DATA C9,AF,32,22,EF,C3,CF,0A,0E,0A,21,22,EF,79,32,C2
340 DATA ED,CD,C3,60,C2,66,10,78,FE,01,9F,20,03,32,C2,ED
350 DATA 21,76,EA,C3,58,10,FE,50,38,15,21,47,60,4F,06,00
360 DATA 09,7E,A7,37,C8,A7,C9,09,1F,1D,00,00,2D,2F,00,CD
370 DATA A1,10,D8,FE,41,38,0C,FE,5B,38,0A,FE,61,38,04,FE
380 DATA 7B,38,02,A7,C9,F5,3A,23,EF,A7,20,04,F1,EE,20,C9
390 DATA F1,A7,C9,3E,0A,B9,C2,80,10,DB,0A,E6,80,32,23,EF
400 DATA 16,7F,C3,89,10,D5,11,68,0A,CD,95,40,20,09,DB,40
410 DATA E6,02,20,03,3E,22,11,3E,02,D3,31,DB,40,E6,02,20
420 DATA 04,11,94,56,19,D1,3A,67,EA,C3,FC,09,B7,8B,98,6F
430 DATA 58,5F,89,93,73,38,F5,3A,55,EA,D3,E4,F1,FB,C9,CD
440 DATA 02,16,C3,3B,17,00,00,00,00,00,00,00,00,00,00,00
450 '
460 POKE   &H84,&HCD
470 POKE   &H85,&H46 : POKE   &H86,&H60
480 POKE  &H9FA,&HD5 : POKE  &H9FB,&H60
490 POKE  &HB18,&HCD
500 POKE  &HB19,&H3F : POKE  &HB1A,&H60
510 POKE  &HFD9,&H68 : POKE  &HFDA,&H60
520 POKE  &HFEC,&H86 : POKE  &HFED,&H60
530 POKE &H1710, &HF : POKE &H1711,&H61
540 POKE &H179F,&H61 : POKE &H17A0,&H60
550 POKE &H17F9, &H6 : POKE &H17FA,&H61
560 POKE &H187E, &H6 : POKE &H187F,&H61
570 POKE &H1880, &H6 : POKE &H1881,&H61
580 POKE &H17DA,&H62
590 POKE &H1850,&H33
600 '
610 DEF USR=&HFF5C : DM=USR(0)
620 '
630 END
```

N-BASICでこのプログラムを実行すると、リセットがかかって、次の画面が表示されます。

```
NEC PC-8001 BASIC Ver 1.3
Copyright 1979 (C) by Microsoft

Ok
```

フリーエリアの大きさを見てみましょう。

```
print fre(0)
 34466
Ok
```

確かに、7.5Kバイト増えていますね。

この例は、DISKがない場合ですが、DISK-BASICモードであってもこのまま使うことができます。ドライブ1にシステムディスクをセットしてこのプログラムを実行して下さい。

```
Two surface disk version [20-Sep-1981]
How many files(0-15)? 3
NEC PC-8001 BASIC Ver 1.3
Copyright 1979 (C) by Microsoft

Ok
print fre(0)
 27029
Ok
```

これで大きなプログラムもDISK-BASICモードで走らせることができるようになります。

≪注意≫

- このモードでは、N-BASICインタプリタがRAM上にあります。もし、この部分が破壊されたりすると暴走する危険がありますので注意して下さい。
- リセットボタンを押すと(ホットスタートも含む)もとのROMバージョンのN-BASICモードに戻ってしまいます。このモードのままでリセットしたい場合（新たなDOSを読み込むときなど）は、モニタモードでG0⏎を実行して下さい。

1-6 すべてのROM,RAMの PEEK,POKEプログラム

今まで述べてきた方法で、すべてのメモリの読み書きができますが、これをBASICで行うわけには行きません。そこで、これをBASICで手軽に行なえるようにする機械語のプログラムを紹介しましょう。

1）まず、clear, &HE3FF⏎を行なってからプログラム1-6の機械語プログラムをモニタで正確に入力して下さい。入力し終わったら必ずセーブして下さい。テープのときはモニタのwコマンド、ディスクのときはBSAVE文で行なうのがよいでしょう。
2）モニタで「ge400⏎」と行い、 CTRL ＋BでBASICに戻ります。
3）これでBASICのPEEK,POKEですべてのメモリの読み書きができるようになりました（ただし、拡張ROM、拡張RAMの読み書きはできません）。

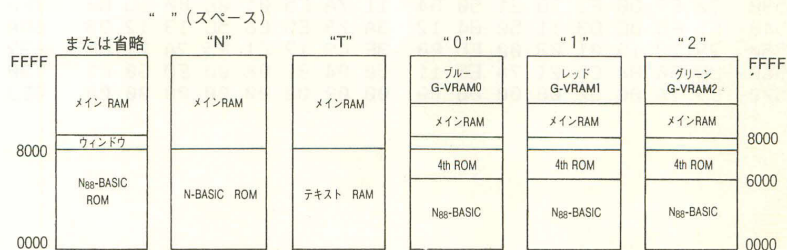
使用方法は次の通りです。

〈PEEK〉

PEEK（アドレス，“バンク名”）

〈POKE〉

POKEアドレス，データ，“バンク名”

アドレスとデータは今までと同じです。バンク名には6つあります。バンク名とその時のメモリ・マップを示します。（図1-6-1）

たとえば、テキストRAMの123H番地を読む時は、PEEK（&H123,"T"）⏎と行ないます。また、POKE &HC000,255,"2"⏎ではグラフィックRAMのグリーンページのデータ&HFFに書き込まれ、画面の左上スミに横棒が現れます。グラフィックRAMと画面の関係は第4章を見て下さい。通常のPEEK文，POKE文はこれまで通り行えます。

ADRS	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	SUM
E400-	3A	74	E5	B7	20	34	F3	21	27	ED	11	74	E5	01	03	00	634
E410-	ED	B0	21	9F	ED	11	77	E5	01	03	00	ED	B0	3E	C3	32	78B
E420-	27	ED	32	9F	ED	3A	27	ED	21	3C	E4	22	28	ED	21	EB	7A4
E430-	E4	22	A0	ED	3E	6C	32	27	E8	FB	FF	C9	3C	28	04	3D	7E6
E440-	C3	74	E5	23	7E	FE	97	28	04	2B	3E	FF	C9	F1	CD	77	8E4
E450-	E5	3E	05	F5	D7	CF	28	CD	93	1B	D5	2B	D7	FE	29	28	88C
E460-	09	CF	2C	CD	BC	E4	D1	C1	F5	D5	CF	29	D1	ED	53	71	A47
E470-	E5	F1	32	73	E5	E5	3E	02	32	BD	EA	2A	71	E5	CD	28	8D3
E480-	E5	CD	99	E4	4E	D3	5F	3E	FF	D3	71	3A	C2	E6	D3	31	A16
E490-	FB	69	26	00	22	41	EC	E1	C9	F3	3A	73	E5	D6	03	30	811
E4A0-	05	3E	FE	D3	71	C9	20	08	3A	C2	E6	F6	02	D3	31	C9	81D
E4B0-	3D	C0	3A	C2	E6	F6	04	E6	FD	D3	31	C9	CD	D3	11	E5	A1F
E4C0-	CD	C9	56	7E	B7	CA	06	0B	23	5E	23	56	1A	0E	05	FE	621
E4D0-	20	28	15	0D	FE	4E	28	10	0D	FE	54	28	0B	D6	30	DA	560
E4E0-	06	0B	FE	03	D2	06	0B	4F	79	E1	C9	F1	CD	77	E5	CF	850
E4F0-	2C	3E	05	F5	CD	A3	18	F5	2B	D7	28	09	CF	2C	CD	BC	798

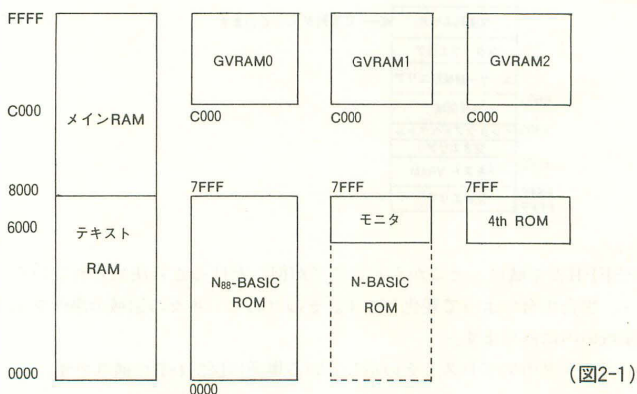
ADRS	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	SUM
E500-	E4	C1	D1	F5	C5	C1	F1	32	73	E5	D1	ED	53	71	E5	E5	BB8
E510-	2A	71	E5	CD	28	E5	CD	99	E4	70	D3	5F	3E	FF	D3	71	9C7
E520-	3A	C2	E6	D3	31	FB	E1	C9	3A	73	E5	FE	03	D0	11	00	8FF
E530-	C0	E7	D8	F3	78	21	50	84	11	7A	E5	01	06	00	ED	B0	7F3
E540-	E1	F5	3E	D3	11	50	84	12	3A	73	E5	C6	5C	13	12	23	6DA
E550-	23	23	13	01	03	00	ED	B0	3E	C9	12	C1	E5	2A	71	E5	639
E560-	CD	50	84	C5	21	7A	E5	11	50	84	01	06	00	ED	B0	C1	730
E570-	C9	1A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0E3

第2章　N_{88}-BASICの内部構造

第2章　N_{88}-BASICの内部構造

2-1. N_{88}-BASICメモリマップ

N_{88}-BASICモードではPC-8801の多くのROM、RAMのうち、図2-1のものを使用します。

(図2-1)

これらのROM、RAMの使用目的は、

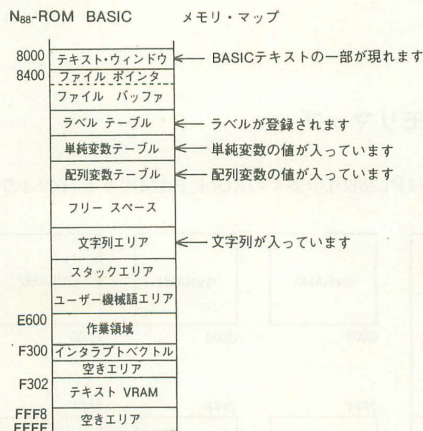
1）N_{88}-BASIC ROM…………………N_{88}-BASICの主要部分
2）モニタ……………………………主にN_{88}-BASICモードのモニタ
3）4th ROM…………………………主にグラフィック関係のルーチン
4）メインRAM………………………変数領域、作業領域、VRAMなど
5）テキストRAM……………………BASICプログラムの格納領域
6）グラフィックVRAM１、２、３…グラフィック表示用

となっています。

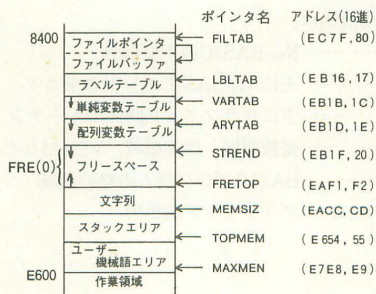
2-1-1 メインRAM

A. N88-ROM BASICメモリ・マップ

N88-ROM BASICではメインRAMは図2-1-Aのように使われています。

（図2-1-A）

8400A～E5FFHの領域は、どこからどこまでが何、とはっきり決められているわけではありません。場合場合によって変化します。そのために、各々の領域の場所を示すポインタが作業領域の内にあります。

それぞれのポインタのアドレスとその示している場所は図2-1-Bの通りです。

（図2-1-B）

これらのポインタのうち、FILTAB，LBLTAB，MAXMEMは、リセット時に決められます。また、MEMSIZ，TOPMEMはCLEAR文によって変更できます。

例えば、CLEAR,&HDFFF,1000⏎と行うと、TOPMEMはDFFFH，MEMSIZはDFFFH−1000=DC17Hとなります。実際にモニタで見てみましょう。

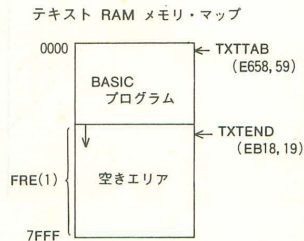
```
clear,&Hdfff,1000
Ok
mon

h]de654
E654 FF DF FF FF 01 00 1C 6F 4E 45 43 30 30 30 30 30
h]deacc
EACC 17 DC D0 EA 05 1A 03 02 36 DA 03 CD 80 00 00 00
h]^b
Ok
```

B. N88-DISK BASICメモリマップ

N88-DISK BASICになると、メインRAM上にDISKのためのルーチン(ディスク・コード)などが置かれるため、メモリマップは図2-1-Cのようになります。

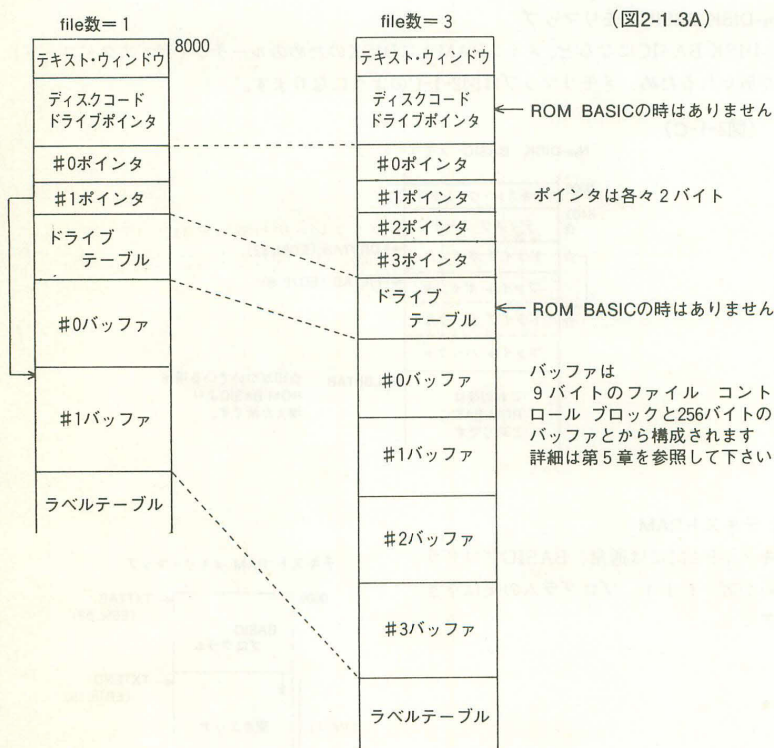
(図2-1-C)

2-1-2 テキストRAM

テキストRAMには通常、BASICプログラムのみが置かれます。プログラムの後は空きエリアです。

2-1-3 メモリ・マップの変化

A. ファイルポインタとファイルバッファ

ファイルポインタとファイルバッファは、電源ON時の「How many files?」に対する答えによって変わってきます。入力した数をNとすると、＃０～＃NのN＋１個のポインタとバッファがとられます。ただ⏎キーを押した時は、ROM BASICのとき２個（１を答えたのと同じ)、DISK BASICのときディスクドライブの数＋１個のポインタとバッファがとられます。この数はEC7EH番地に入っています。EC7EH番地が３のときは＃０～＃３の４個のポインタとバッファがあるということです。ただし、＃０のバッファは普通の用途には使えません。DSKI$，DSKO$などに使われます。FIELD＃０は可能ですが、OPEN～AS＃０はできません。

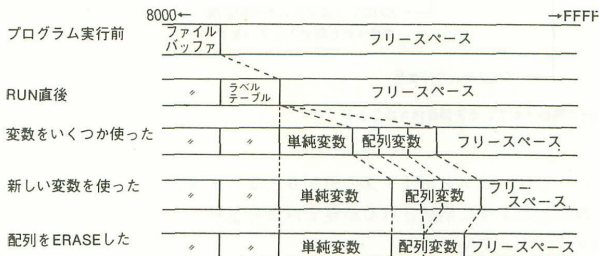
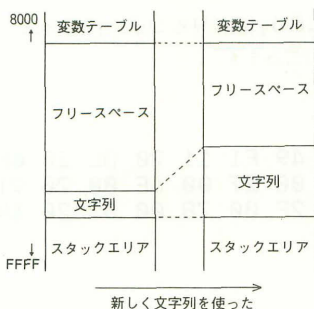
図2-1-3Aはオープンできるファイル数が１個の場合と３個の場合のメモリ・マップを示します。

B. ラベル・変数テーブル

ラベル・変数テーブルは、プログラムによって、またその実行中に刻々と変化します。

ラベルテーブルは、RUNの最初，CLEAR，LOAD後などの時にプログラム中からラベルを集めて、一気に作られます。この時、この後の変数テーブルは消されてしまいます。またRUNの時は最初にテーブルが作られてしまうわけですから、実行中にラベルテーブルが変化することはありません。

これに対して変数テーブルは、プログラム実行中に新しい変数が現われるたびに変化します。その概念図を図2-1-3Bに示します。

(図2-1-3B)

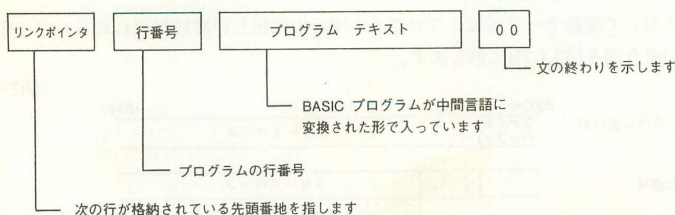
	8000←				→FFFF
プログラム実行前	ファイルバッファ	フリースペース			
RUN直後	〃	ラベルテーブル	フリースペース		
変数をいくつか使った	〃	〃	単純変数	配列変数	フリースペース
新しい変数を使った	〃	〃	単純変数	配列変数	フリースペース
配列をERASEした	〃	〃	単純変数	配列変数	フリースペース

C. 文字列領域

(図2-1-3C)

図2-1-3Cのように、文字列はメモリの後のアドレスから順に格納されていきます。このとき、同じ文字変数に新しい文字列を代入すると、古い文字列はそのままで、新たに文字列領域を増やして、そこに新しい文字列を書き込みます。そのため、文字列領域内には、もう使われていない、ゴミとなった文字列がたまっていきます。これはどうするのかというと、いずれ文字列領域が増えてフリースペースがなくなった時、新しい文字列を作ろうとしても作る所がありませんから、その時にこれらのゴミを処分します。ゴミの部分を消して、使用中の文字列を後から順番につめていくわけです。こうして新しい文字列のためのスペースを作ります。これをガーベッジ・コレクション（garbage collection）と呼びます。多くの文字列を作ったプログラムを実行していると、しばらく実行が止まってしまうことがありますが、それはこのガーベッジ・コレクションを行っているためです。また、DIM文で配列を宣言した時にフリースペースが足りない時もガーベッジ・コレクションを行って、必要なメモリを確保しようとします。なお、N_{88}-BASICリファレンスマニュアルにもある通り、FRE関数でもガーベッジ・コレクションを行います。

2-2. プログラムの格納状態

BASICプログラムは、テキストRAMの先頭から、図2-2-Aのような形式で格納されていきます。

(図2-2-A)

プログラムの終わりでは、リンクポインタの値が0となります。
次に、プログラムがどのように格納されるか見てみましょう。
例として、次のプログラムを使います。

```
100 FOR I=0 TO 20 STEP .5
110  PRINT I,SQR(I)
120 NEXT I
```

テキストRAMは、BASIC ROMと同じ番地にあり、直接見ることができません。
そこで、テキスト・ウィンドウを使ってのぞいてみます。

```
h]o70,0
h]d8000,8030
8000 00 18 00 64 00 82 20 49 F1 11 20 DC 20 0F 14 20
8010 DF 20 1D 00 00 00 80 00 27 00 6E 00 20 91 20 49
8020 2C FF 87 28 49 29 00 2F 00 78 00 83 20 49 00 00
8030 00
h]
```

※h] o70,0とあるのは、0H番地から3FFH番地をテキスト・ウィンドウに映すために、ポート70Hに、データ0をOUTする命令です。

各データは、図2-2-Bのような意味を持ちます。

(図2-2-B)

アドレス	データ	意味
0001	18	リンクポインター→
0002	00	
0003	64	行番号(=100)
0004	00	
0005	82	FOR
0006	20	スペース
0007	49	I
0008	F1	=
0009	11	0
000A	20	スペース
000B	DC	TO
000C	20	スペース
000D	0F	20
000E	14	
000F	20	スペース
0010	DF	STEP
0011	20	スペース
0012	1D	0.5
0013	00	
0014	00	
0015	00	
0016	80	
0017	00	行の終わり
0018	27	リンクポインター→
0019	00	
001A	6E	行番号(=110)
001B	00	
001C	20	スペース
001D	91	PRINT
001E	20	スペース
001F	49	I
0020	2C	,
0021	FF	SQR
0022	87	
0023	28	(
0024	49	I
0025	29	)
0026	00	行の終わり
0027	2F	リンクポインター→
0028	00	
0029	78	行番号(=120)
002A	00	
002B	83	NEXT
002C	20	スペース
002D	49	I
002E	00	行の終わり
002F	00	プログラムの終わり
0030	00	

2-3. 中間言語

前の節で、BASICプログラムのテキストが中間言語を使って、短縮された形で格納されていることがわかりました。

それでは、N88-BASICでどのような中間言語が使われているのかを見てみましょう。

2-3-1 中間言語コード（0H～7FH）

中間言語コードの0Hから7FHは、数値や行番号、変数名などに使われます。

中間言語		意味	備考
0A	LF	(Line Feed) CTRL ＋ J で入力	
0B	&O	以下の2バイトは8進数	0B 9C 02＝&O1234
0C	&H	以下の2バイトは16進数	0C 34 12＝&H1234
0D	アドレス	以下の2バイトは飛び先絶対アドレス	GOTO, GOSUB, THEN ELSE, RESTOREの後に続きます。
0E	行番号	以下の2バイトは飛び先行番号	
0F	整数(1バイト)	以下の1バイトは、10～255の整数	0F 50＝80
11～1A	整数(1ケタ)	1桁の整数，(11→0，12→1，1A→9)	
1B*		整　数　10	
1C	整数(2バイト)	以下の2バイトは，整数	1C D2 04＝1234
1D	単精度(4バイト)	以下の4バイトは，単精度定数	1D EB C0 1D 81 ＝1.2345
1F	倍精度(8バイト)	以下の8バイトは，倍精度定数	
20 ～ 7F	文字	キャラクタコードに対応する文字（変数名やラベル名など）	アルファベットの小文字は大文字に変換されるため使われません。

(*) 通常使われていません。

(図 2-3-1)

2-3-2 中間言語コード（80H～FFH）

中間言語コードの80HからFFHは、1バイトまたは2バイトで、N88-BASICのキーワードを示します。(表2-3-1～2)

N-BASICでも似たようなキーワードと中間言語コードを持っていますが、N88-BASICと比べると対応していなかったり、バイト数が違っているものもあります。N-BASICのプログラムを「LOAD“CAS2:ファイル名”」でロードして、リストをとってみるとこの違いがわかるでしょう。

(1)1バイトで表わされるもの

コード		コード		コード	
80	:	B0	: WEND	E0	: USR
81	: END	B1	: CALL	E1	: FN
82	: FOR	B2	:	E2	: SPC(
83	: NEXT	B3	:	E3	: NOT
84	: DATA	B4	:	E4	: ERL
85	: INPUT	B5	: WRITE	E5	: ERR
86	: DIM	B6	: COMMON	E6	: STRING$
87	: READ	B7	: CHAIN	E7	: USING
88	: LET	B8	: OPTION	E8	: INSTR
89	: GO TO	B9	: RANDOMIZE	E9	: ´
8A	: RUN	BA	: DSKO$	EA	: VARPTR
8B	: IF	BB	: OPEN	EB	: ATTR$
8C	: RESTORE	BC	: FIELD	EC	: DSKI$
8D	: GOSUB	BD	: GET	ED	: SRQ
8E	: RETURN	BE	: PUT	EE	: OFF
8F	: REM	BF	: SET	EF	: INKEY$
90	: STOP	C0	: CLOSE	F0	: >
91	: PRINT	C1	: LOAD	F1	: =
92	: CLEAR	C2	: MERGE	F2	: <
93	: LIST	C3	: FILES	F3	: +
94	: NEW	C4	: NAME	F4	: −
95	: ON	C5	: KILL	F5	: *
96	: WAIT	C6	: LSET	F6	: /
97	: DEF	C7	: RSET	F7	: ^
98	: POKE	C8	: SAVE	F8	: AND
99	: CONT	C9	: LFILES	F9	: OR
9A	: OUT	CA	: MON	FA	: XOR
9B	: LPRINT	CB	: COLOR	FB	: EQV
9C	: LLIST	CC	: CIRCLE	FC	: IMP
9D	: CONSOLE	CD	: COPY	FD	: MOD
9E	: WIDTH	CE	: CLS	FE	: ¥
9F	: ELSE	CF	: PSET	FF	:
A0	: TRON	D0	: PRESET		
A1	: TROFF	D1	: PAINT		
A2	: SWAP	D2	: TERM		
A3	: ERASE	D3	: SCREEN		
A4	: EDIT	D4	: BLOAD		
A5	: ERROR	D5	: BSAVE		
A6	: RESUME	D6	: LOCATE		
A7	: DELETE	D7	: BEEP		
A8	: AUTO	D8	: ROLL		
A9	: RENUM	D9	: HELP		
AA	: DEFSTR	DA	:		
AB	: DEFINT	DB	: KANJI		
AC	: DEFSNG	DC	: TO		
AD	: DEFDBL	DD	: THEN		
AE	: LINE	DE	: TAB(		
AF	: WHILE	DF	: STEP		

(2) 2バイトで表わされるもの

コード		命令
FF 80	:	
FF 81	:	LEFT$
FF 82	:	RIGHT$
FF 83	:	MID$
FF 84	:	SGN
FF 85	:	INT
FF 86	:	ABS
FF 87	:	SQR
FF 88	:	RND
FF 89	:	SIN
FF 8A	:	LOG
FF 8B	:	EXP
FF 8C	:	COS
FF 8D	:	TAN
FF 8E	:	ATN
FF 8F	:	FRE
FF 90	:	INP
FF 91	:	POS
FF 92	:	LEN
FF 93	:	STR$
FF 94	:	VAL
FF 95	:	ASC
FF 96	:	CHR$
FF 97	:	PEEK
FF 98	:	SPACE$
FF 99	:	OCT$
FF 9A	:	HEX$
FF 9B	:	LPOS
FF 9C	:	CINT
FF 9D	:	CSNG
FF 9E	:	CDBL
FF 9F	:	FIX
FF A0	:	CVI
FF A1	:	CVS
FF A2	:	CVD
FF A3	:	EOF
FF A4	:	LOC
FF A5	:	LOF
FF A6	:	FPOS
FF A7	:	MKI$
FF A8	:	MKS$
FF A9	:	MKD$
FF AA	:	
FF AB	:	
FF AC	:	
FF AD	:	
FF AE	:	
FF AF	:	
FF B0	:	
FF B1	:	
FF B2	:	
FF B3	:	
FF B4	:	
FF B5	:	
FF B6	:	
FF B7	:	
FF B8	:	
FF B9	:	
FF BA	:	
FF BB	:	
FF BC	:	
FF BD	:	
FF BE	:	
FF BF	:	
FF C0	:	
FF C1	:	
FF C2	:	
FF C3	:	
FF C4	:	
FF C5	:	
FF C6	:	
FF C7	:	
FF C8	:	
FF C9	:	
FF CA	:	
FF CB	:	
FF CC	:	
FF CD	:	
FF CE	:	
FF CF	:	
FF D0	:	DSKF
FF D1	:	VIEW
FF D2	:	WINDOW
FF D3	:	POINT
FF D4	:	CSRLIN
FF D5	:	MAP
FF D6	:	SEARCH
FF D7	:	MOTOR
FF D8	:	PEN
FF D9	:	DATE$
FF DA	:	COM
FF DB	:	KEY
FF DC	:	TIME$
FF DD	:	WBYTE
FF DE	:	RBYTE
FF DF	:	POLL
FF E0	:	ISET
FF E1	:	IEEE
FF E2	:	IRESET
FF E3	:	STATUS
FF E4	:	CMD
FF E5	:	
FF E6	:	
FF E7	:	
FF E8	:	
FF E9	:	
FF EA	:	
FF EB	:	
FF EC	:	
FF ED	:	
FF EE	:	
FF EF	:	
FF F0	:	
FF F1	:	
FF F2	:	
FF F3	:	
FF F4	:	
FF F5	:	
FF F6	:	
FF F7	:	
FF F8	:	
FF F9	:	
FF FA	:	
FF FB	:	
FF FC	:	
FF FD	:	
FF FE	:	
FF FF	:	

2-3-3 中間言語テーブル

中間言語とキーワードの対応表は、ROM内の6B8AH番地から6E95H番地に格納されています。

このテーブルは、入力したプログラムを中間言語に変換してテキスト領域に格納するときや、逆にリストをとったりするときに使われるものです。

• 中間言語テーブルの形成

中間言語はキーワードの1文字目によって、アルファベット順に分類されています。

分類されたキーワードの格納領域を示すテーブルが、6B56H番地から6B89H番地にあります。

```
A  ------  6B8AH        N  ------  6D40H
B  ------  6BA0H        O  ------  6D4FH
C  ------  6BAFH        P  ------  6D68H
D  ------  6C0EH        Q  ------  6D97H
E  ------  6C4AH        R  ------  6D98H
F  ------  6C6FH        S  ------  6DDAH
G  ------  6C89H        T  ------  6E21H
H  ------  6C9BH        U  ------  6E41H
I  ------  6CA4H        V  ------  6E4AH
J  ------  6CCEH        W  ------  6E58H
K  ------  6CCFH        X  ------  6E7BH
L  ------  6CDCH        Y  ------  6E7FH
M  ------  6D1CH        Z  ------  6E80H
```

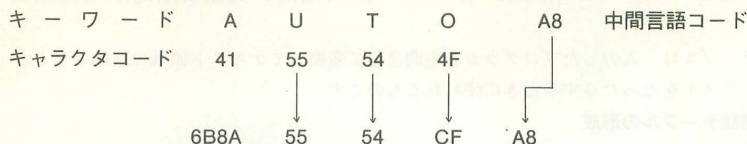
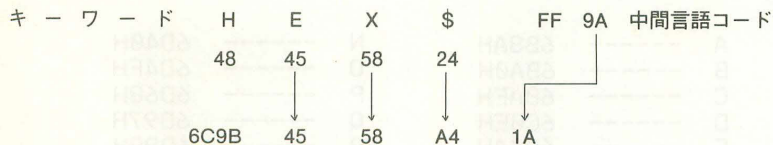
また、このテーブルを参照しなくても、各グループの間には、セパレータとして00が書き込まれています。

例

```
ATTR$                    BSAVE
54  54  52  A4  EB  [00]  53  41  56
```

次に、中間言語テーブルのデータ構造を見てみましょう。ここでも、メモリ節約のためいろいろな工夫がなされています。

各データは、キーワードと中間言語とから成り立っています。キーワードのデータは1文字目が省略され（1文字目でグループ分けしてあるので不要)、キーワードの最後を示すために、最後の文字データの最上位ビットを1にしてあります。また、中間言語コードについては、1バイトのものはそのままの形で、2バイトのもの（FF+XX）は、最上位ビットを0にして1バイトで表わせるようにしてあります。

〈中間言語コードが１バイト〉

〈中間言語コードが２バイト〉

次のプログラムで、これらのデータ（キーワードと中間言語）を出力してみましょう。

```
100 '
110 '----- N88-BASIC Key Word List ----- [ 1 ]
120 '
130  KW.TBL=&H6B8A
140  TOP.WORD=ASC("A")
150 '
160 *SEARCH.KW
170  KEY.WORD$=CHR$(TOP.WORD)
180 *NEXT.CODE
190  GOSUB *GET.CODE
200  IF CODE=0 THEN TOP.WORD=TOP.WORD+1 : GOTO *SEARCH.KW
210  IF CODE<&H80 THEN KEY.WORD$=KEY.WORD$+CHR$(CODE) : GOTO *NEXT.CODE
220 '
230  KEY.WORD$=KEY.WORD$+CHR$(CODE AND &H7F)
235  IF TOP.WORD=ASC("Z")+1 THEN KEY.WORD$=MID$(KEY.WORD$,2)
240  PRINT TAB(10);KEY.WORD$;
250 '
260  GOSUB *GET.CODE
270  PRINT TAB(20);
280  IF CODE<&H80 THEN PRINT "FF ";
290  PRINT RIGHT$("0"+HEX$(CODE OR &H80),2)
300 '
310  IF KW.TBL<&H6E95 THEN *SEARCH.KW
320 '
330  END
340 '
350 *GET.CODE
360  CODE=PEEK(KW.TBL)
370  KW.TBL=KW.TBL+1
380  RETURN
```

Keyword	Code
AUTO	A8
AND	F8
ABS	FF 86
ATN	FF 8E
ASC	FF 95
ATTR$	EB
BSAVE	D5
BLOAD	D4
BEEP	D7
CONSOLE	9D
COPY	CD
CLOSE	C0
CONT	99
CLEAR	92
CSRLIN	FF D4
CINT	FF 9C
CSNG	FF 9D
CDBL	FF 9E
CVI	FF A0
CVS	FF A1
CVD	FF A2
COS	FF 8C
CHR$	FF 96
CALL	B1
COMMON	B6
CHAIN	B7
COM	FF DA
CIRCLE	CC
COLOR	CB
CLS	CE
CMD	FF E4
DELETE	A7
DATA	84
DIM	86
DEFSTR	AA
DEFINT	AB
DEFSNG	AC
DEFDBL	AD
DSKO$	BA
DEF	97
DSKI$	EC
DSKF	FF D0
DATE$	FF D9
ELSE	9F
END	81
ERASE	A3
EDIT	A4
ERROR	A5
ERL	E4
ERR	E5
EXP	FF 8B
EOF	FF A3
EQV	FB
FOR	82
FIELD	BC
FILES	C3
FN	E1
FRE	FF 8F
FIX	FF 9F
FPOS	FF A6
GOTO	89
GO TO	89
GOSUB	8D
GET	BD
HEX$	FF 9A
HELP	D9
INPUT	85
ISET	FF E0
IEEE	FF E1
IRESET	FF E2
IF	8B
INSTR	E8
INT	FF 85
INP	FF 90
IMP	FC
INKEY$	EF
KEY	FF DB
KILL	C5
KANJI	DB
LOCATE	D6
LPRINT	9B
LLIST	9C
LPOS	FF 9B
LET	88
LINE	AE
LOAD	C1
LSET	C6
LIST	93
LFILES	C9
LOG	FF 8A
LOC	FF A4
LEN	FF 92
LEFT$	FF 81
LOF	FF A5
MOTOR	FF D7
MERGE	C2
MOD	FD
MKI$	FF A7
MKS$	FF A8
MKD$	FF A9
MID$	FF 83
MON	CA
MAP	FF D5
NEXT	83
NAME	C4
NEW	94
NOT	E3
OPEN	BB
OUT	9A
ON	95
OR	F9
OCT$	FF 99
OPTION	B8
OFF	EE
PRINT	91
PUT	BE
POKE	98
POLL	FF DF
POS	FF 91
PEEK	FF 97
PSET	CF
PRESET	D0
POINT	FF D3
PAINT	D1
PEN	FF D8
RETURN	8E
READ	87
RUN	8A
RESTORE	8C
RBYTE	FF DE
REM	8F
RESUME	A6
RSET	C7
RIGHT$	FF 82
RND	FF 88
RENUM	A9
RANDOMIZE	B9
ROLL	D8
SCREEN	D3
SEARCH	FF D6
STOP	90
SWAP	A2
SET	BF
SRQ	ED
STATUS	FF E3
SAVE	C8
SPC(	E2
STEP	DF
SGN	FF 84
SQR	FF 87
SIN	FF 89
STR$	FF 93
STRING$	E6
SPACE$	FF 98
THEN	DD
TRON	A0
TROFF	A1
TAB(	DE
TO	DC
TAN	FF 8D
TERM	D2
TIME$	FF DC
USING	E7
USR	E0
VAL	FF 94
VIEW	FF D1
VARPTR	EA
WIDTH	9E
WINDOW	FF D2
WAIT	96
WHILE	AF
WEND	B0
WRITE	B5
WBYTE	FF DD
XOR	FA
+	F3
−	F4
*	F5
/	F6
^	F7
¥	FE
′	E9
>	F0
=	F1
<	F2

また、前のプログラムを多少変更するだけで、同様の表が得られます。

```
100 '
110 '----- N88-BASIC Key Word List ----- [ 2 ]
120 '
130  DIM K.W$(255)
140 '
150  KW.TBL=&H6B8A
160  TOP.WORD=ASC("A")
170 '
180 *SEARCH.KW
190  KEY.WORD$=CHR$(TOP.WORD)
200 *NEXT.CODE
210  GOSUB *GET.CODE
220  IF CODE=0 THEN TOP.WORD=TOP.WORD+1 : GOTO *SEARCH.KW
230  IF CODE<&H80 THEN KEY.WORD$=KEY.WORD$+CHR$(CODE) : GOTO *NEXT.CODE
240 '
250  KEY.WORD$=KEY.WORD$+CHR$(CODE AND &H7F)
260 '
270  GOSUB *GET.CODE
280  IF TOP.WORD=ASC("Z")+1 THEN KEY.WORD$=MID$(KEY.WORD$,2)
290  K.W$(CODE)=KEY.WORD$
300 '
310  IF KW.TBL<&H6E95 THEN *SEARCH.KW
320 '
330  FOR CODE=128 TO 255
340    PRINT HEX$(CODE)" : ";K.W$(CODE)
350  NEXT
360  FOR CODE=0 TO 127
370    PRINT "FF "HEX$(CODE+&H80)" : ";K.W$(CODE)
380  NEXT
390 '
400  END
410 '
420 *GET.CODE
430  CODE=PEEK(KW.TBL)
440  KW.TBL=KW.TBL+1
450  RETURN
```

2-3-4 リストでBEEP音を

中間言語の１から９は普通使われませんが、この中でおもしろい使い方ができるのが、コード７です。

これはBELコード（BEEP音）を表わし、直接キーボードから入力することはできません。

そこで、直接、中間コードを書き直すことにしましょう。

次のプログラムを入力します。（'＊'は、BEEP音を出したいところを示します。）

```
list
10 REM *S*A*M*P*L*E*
Ok
```

次に、モニタモードで、必要な部分＊（コード2AHのところ）をコード７に直します。

```
mon

h]d8000,8016
8000 00 15 00 0A 00 8F 20 2A 53 2A 41 2A 4D 2A 50 2A        ␊ ┗ ┼ *S*A*M*P*
8010 4C 2A 45 2A 00 00 00                                   L*E*
h]s8007
8007 2A-07 53- 2A-07 41- 2A-07 4D- 2A-07 50- 2A-07 4C- 2A-07 45- 2A-07 00-
8015 00- 00-
h]d8000,8016
8000 00 15 00 0A 00 8F 20 07 53 07 41 07 4D 07 50 07        ␊ ┗ ┼ ␇S␇A␇M␇P␇
8010 4C 07 45 07 00 00 00                                   L␇E␇
h]^b
Ok
```

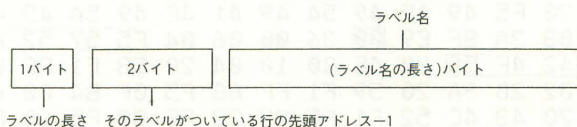
BASICに戻して、リストをとってみて下さい。どうですか、1文字毎にBEEP音が出ますね。

```
list
10 REM SAMPLE
Ok
```

なお、この行を修正する（修正しなくても、この行のところで⏎キーを押す）と、リストをとっても音は出なくなります。

2-4. ラベルテーブル

ラベルテーブルは、LBLTAB（EB16,17）で示されるアドレスから、VARTAB（EB1B,1C）で示されるアドレスの1つ前までです。

ラベルは、この中に、次の様な形式で登録されます。

それでは、実際にラベルがどのような形で登録されるか見てみましょう。

次のプログラムを入力した後、CLEAR文を実行します。

```
1000 *START
1010  GOSUB *INITIALIZE
1020 '
1030 *WRITE.BOX
1040  X=RND*600 : Y=RND*180
1050  CLR=INT(RND*7+1)
1060  LINE(X,Y)-STEP(40,20),CLR,B
1070  PAINT(X+20,Y+10),INT(RND*7+1),CLR
1080 GOTO *WRITE.BOX
1090 '
1100 *INITIALIZE
1110  SCREEN 0,0
1120  RANDOMIZE
1130  CLS 3
1140 RETURN
```

これでラベルテーブルができあがりました。各ポインタの値を見てみましょう。

LBLTAB　　ＥＢ１６　E1　B2

VARTAB　　ＥＢ１Ｂ　02　B3

ラベルテーブルは、B2E1H番地からB301H番地にあるわけですね。それではそちらの方をプログラムと比較して見てみます。

• ラベル・テーブル

• テキストウィンドウから見たプログラムデータ（実際は０番地から）

```
8000 00 0C 00 E8 03 F5 53 54 41 52 54 00 1F 00 F2 03
8010 20 8D 20 F5 49 4E 49 54 49 41 4C 49 5A 45 00 27
8020 00 FC 03 3A 8F E9 00 36 00 06 04 F5 57 52 49 54
8030 45 2E 42 4F 58 00 4E 00 10 04 20 58 F1 FF 88 F5
8040 1C 58 02 20 3A 20 59 F1 FF 88 F5 0F B4 00 62 00
8050 1A 04 20 43 4C 52 F1 FF 85 28 FF 88 F5 18 F3 12
8060 29 00 7D 00 24 04 20 AE 28 58 2C 59 29 F4 DF 28
8070 0F 28 2C 0F 14 29 2C 43 4C 52 2C 42 00 9E 00 2E
8080 04 20 D1 28 58 F3 0F 14 2C 59 F3 0F 0A 29 2C FF
8090 85 28 FF 88 F5 18 F3 12 29 2C 43 4C 52 00 AF 00
80A0 38 04 89 20 F5 57 52 49 54 45 2E 42 4F 58 00 B7
80B0 00 42 04 3A 8F E9 00 C7 00 4C 04 F5 49 4E 49 54
80C0 49 41 4C 49 5A 45 00 D2 00 56 04 20 D3 20 11 2C
80D0 11 00 D9 00 60 04 20 B9 00 E2 00 6A 04 20 CE 20
80E0 14 00 E8 00 74 04 8E 00 00 00 00 00 00 00 00 00
```

※オープンするファイルの数によってラベル・テーブルの位置は変ってきますので値が異なる場合があります。

2-5. 変数テーブル

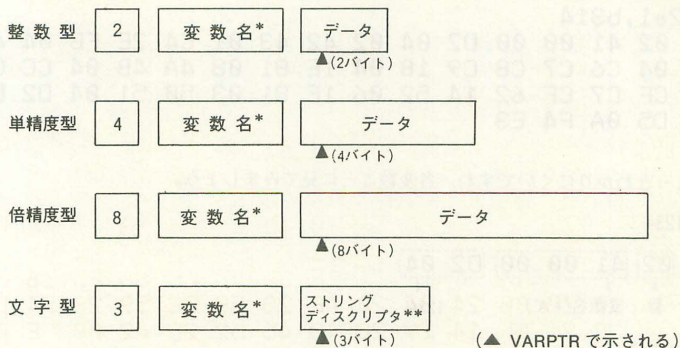
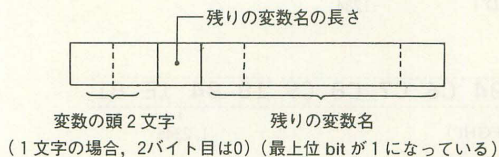
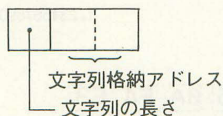
2-5-1 単純変数テーブル

単純変数テーブルは、ラベルテーブルの後に作られます。

この領域は、VARTAB（EB1B,1C）で示されるアドレスから、ARYTAB（EB1D,1E）で示されるアドレスの1つ前までです。

プログラム中で変数が使われると、その型に応じた形式で、それが使われた順番に登録されていきます。変数の値の参照は、このテーブルの先頭から行なわれていきますので、頻繁に使う変数を早めに定義しておくと実行速度を上げることができるわけです。

各変数は、その型によって次の様な形式で登録されます。

単純変数テーブルの構造

*）変数名

**）ストリングディスクリプタ

実際の例で確かめてみます。

次のプログラムを実行した後、各ポインタ及び単純変数テーブルを見てみましょう。

```
100 A%=1234
110 BCD%=-1234
120 DEFGHI!=1.2345
130 JKLMNO#=1.234567890123456#
140 PQRSTU$="1234567890"
```

※ポインタの値

```
h]deb1b
EB1B E1 B2 15 B3
     VARTAB ARYTAB
```

※単純変数テーブル

```
h]db2e1,b314
B2E1 02 41 00 00 D2 04 02 42 43 01 C4 2E FB 04 44 45
B2F1 04 C6 C7 C8 C9 18 04 1E 81 08 4A 4B 04 CC CD CE
B301 CF C7 CF 62 14 52 06 1E 81 03 50 51 04 D2 D3 D4
B311 D5 0A F4 E3
```

ちょっとわかりにくいですね。各変数ごとに見てみましょう。

• A%=1234

```
B2E1 02 41 00 00 D2 04
```

02：型　41 00 00：変数名(='A')　D2 04：1234

• BCD%=−1234

```
B2E7 02 42 43 01 C4 2E FB
```

02：型　42 43 01 C4：変数名(='BCD')　2E FB：−1234

• DEFGH！=1,2345

```
B2EF 04 44 45 04 C6 C7 C8 C9 18 04 1E 81
```

04：型　44 45 04 C6 C7 C8 C9：変数名(='DEFGHI')　18 04 1E 81：1,2345

• JKLMNO＃=1,234567890123456＃

```
B2FA 08 4A 4B 04 CC CD CE CF C7 CF 62 14 52 06 1E 81
```

08：型　4A 4B 04 CC CD CE CF：変数名(='JKLMNO')　C7 CF 62 14 52 06 1E 81：1,234567890123456

• PQRSTU$="1234567890"

```
B30A 03 50 51 04 D2 D3 D4 D5 0A F4 E3
```

03：型　50 51 04 D2 D3 D4 D5：変数名(='PQRSTU')　0A F4 E3：ストリング・ディスクリプタ

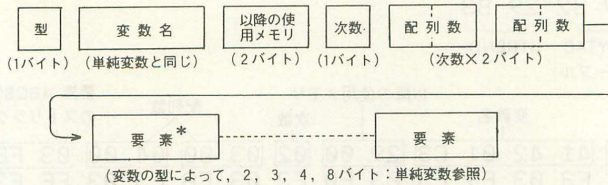
［文字列の長さ=10
文字列の格納アドレス
=E3F4H］

2-5-2 配列変数テーブル

配列変数テーブルは、単純変数テーブルの後に作られます。

このテーブルは、ARYTAB（EB1D,1E）で示されるアドレスから、STREND（EB1F,20）で示されるアドレスの１つ前までです。

ここも、単純変数テーブルと同じように、DIM文を実行したり、添字が10以下の配列変数を使ったときに、その順番で登録されます。ERASE文を実行すると、その配列変数のテーブルが消されて後ろにあるテーブルが前に移動してきます。

配列変数テーブルの形は、各配列について次の様になります。

*）要素の順番は DIM A（2，3，4）の場合、次のようになる。
(OPTION BASE 0 場合)

```
A(0, 0, 0)
A(1, 0, 0)
A(2, 0, 0)
A(0, 1, 0)
A(1, 1, 0)
    ⋮
A(1, 3, 4)
A(2, 3, 4)
```

注）配列の宣言と逆順になる

整数型配列変数と文字型配列変数について実際に見てみましょう。

・整数型配列変数

```
100 DIM A%(3,2)
110 FOR I=0 TO 3
120   FOR J=0 TO 2
130     A%(I,J)=I+J
140   NEXT J
150 NEXT I
```

※ポインタの値

```
h]deb1d
EB1D F1 B2 14 B3
     ARYTAB STREND
```

※配列変数テーブル

```
      型  変数名    以降の使用メモリ  次数  配列数      要素 A%(0,0)
B2F1 [02][41 00 00][1D 00][02][03 00][04 00][00 00] 01 00 02
B301 00 03 00 01 00 02 00 03 00 04 00 02 00 03 00 04
B311 00 [05 00]
        要素 A%(3,2)
```

・文字型配列変数

```
100 DIM ABC$(3,2)
110 FOR I=0 TO 3
120   FOR J=0 TO 2
130     ABC$(I,J)="ABC"
140   NEXT J
150 NEXT I
```

※ポインタの値

```
h]deb1d
EB1D F9 B2 29 B3
     ARYTAB STREND
```

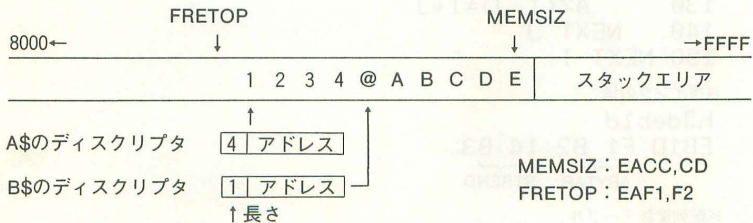
※配列変数テーブル

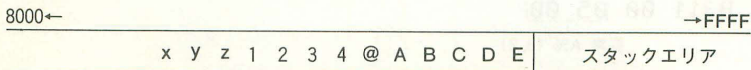
```
     型  変数名       以降の使用メモリ 次数 配列数          要素 ABC$(0,0)のストリングディスクリプタ
B2F9 03 41 42 01 C3 29 00 02 03 00 04 00 03 FB E3 03
B309 F2 E3 03 E9 E3 03 E0 E3 03 F8 E3 03 EF E3 03 E6
B319 E3 03 DD E3 03 F5 E3 03 EC E3 03 E3 E3 03 DA E3
                                             要素 ABC$(3,2)のストリングディスクリプタ
```

2-6. 文字列エリア

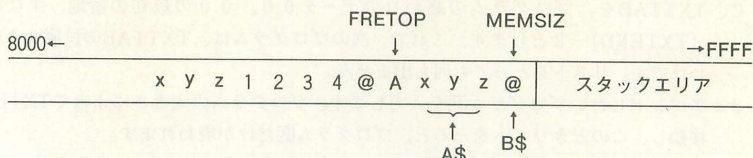
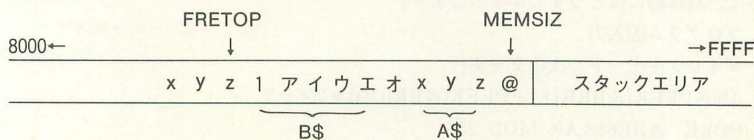
文字列エリアには文字型変数の実際の文字列が納められています。
例として次のプログラムを実行します。

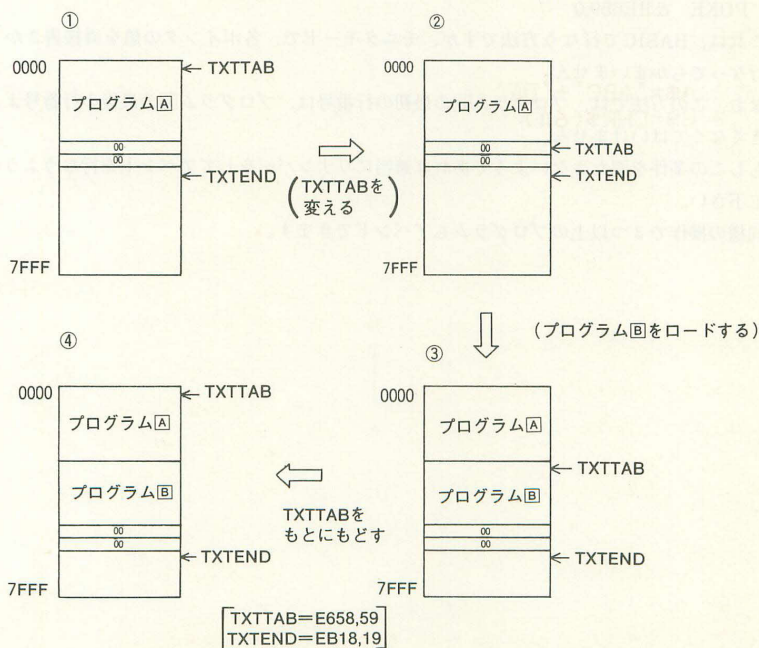
```
10 A$="ABC"+"DE"
20 B$=CHR$(64)
30 A$="1234"
```

このとき文字列エリアは次のようになっています。

「2-1-3 メモリ・マップの変化」で述べた通り、最初にA$に代入された「ABCDE」はメモリ上から消えずにそのまま残っているのです。ここでさらにダイレクトモードでA$＝"xyz"と行なうと次のようになります。

```
8000←                                          →FFFF
            x y z 1 2 3 4 @ A B C D E | スタックエリア
```

ここでガーベッジコレクションを行ってみましょう。

? FRE (0) ↵

こうすると次のようになります。

現在使用されている「xyz」と「@」だけが、文字列エリアの後からつめられて、FRETOPが移動しました。ここでB$＝“アイウ”＋“エオ”と行なうと、このようになります。

2-7. プログラムのアペンド

ここでは、BASICプログラムの簡単なアペンド方法について説明しましょう。

1．もとになるプログラムを入力します。

このとき、TXTTABは、プログラムの先頭番地、TXTENDは、プログラムの終わりを示す、データ00，00の次の番地を指しています。

2．TXTTABを、プログラムの終わりのデータ００，００の最初の番地、すなわち、（TXTEND）－２とします。これで、次のプログラムは、TXTTABの位置からはいるわけです。リストをとっても何も出ません。

3．アペンドしたいプログラム🄱を入力します。プログラムの大きさによってTXTENDが移動し、このときリストをとると、プログラム🄱だけが現われます。

4．TXTTABをもとの値に戻します。これで２つのプログラムがアペンドされました。

さて、具体的にはどうするかを示します。

1．プログラム🄰入力

2．ダイレクトモードで次の文を実行

```
A%=PEEK(&HEB18)+PEEK(&HEB19)*256-2
POKE  &HE658,A% MOD 256
POKE  &HE659,A%¥256
```

3．プログラム🄱入力

4．ダイレクトモードで次の文を実行

```
POKE  &HE658,1
POKE  &HE659,0
```

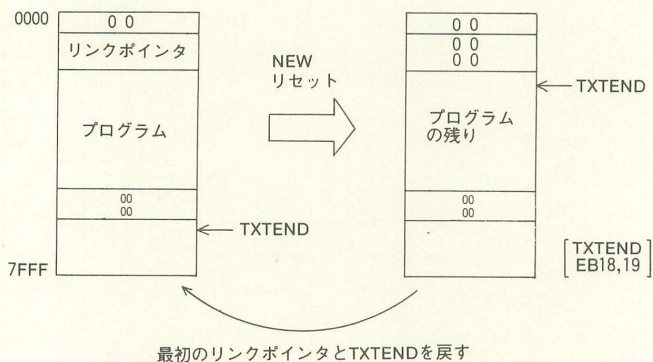
これは、BASICで行なう方法ですが、モニタモードで、各ポインタの値を直接書きかえて行なってもかまいません。

なお、この方法では、プログラム🄱の最初の行番号は、プログラム🄰の最後の行番号より大きくなくてはいけません。

もしこの条件を満たさないようであれば適当にリナンバーをしてアペンドを行なうようにして下さい。

同様の操作で３つ以上のプログラムもアペンドできます。

2-8. BASICプログラム復活

Newやリセットをした場合、以前入っていたプログラムは消えてしまうわけですが、実際にメモリから抹消されてしまうわけではなく、プログラムの最初のリンクポインタとプログラムの終わりを示すポインタ(TXTEND EB18, 19)がクリアされるだけです。従ってこれらの値を元に戻してやればプログラムが復活します。

これを自動的に行うプログラムを次に紹介します。

```
F260 2A 58 E6 36 01 CD BD 05 CD BB 1B CD D5 44 23 22
F270 18 EB AF 32 1A EB FF
```

NEWやリセットした直後にモニタに入り、上のプログラムを正確に打ち込みます。F260H番地から実行させると(GF260☑) すぐに戻ってきますから、BASICに戻り、CLEAR☑とします。これでプログラムが復活しました。

この方法は、ディスクを起動せずに作ったプログラムをディスクに入れる時にも使えます。リセットしてディスクを起動した後に、これを実行すればよいわけです。

第3章　テキスト画面

第3章　テキスト画面

3-1．WIDTHとVRAM

3-1-1 WIDTHとDIPスイッチ

テキスト画面のモードには次の４つの組み合わせがあり、WIDTH文やDIPスイッチ（本体後部にあるSW1）によって設定されます。（図３-１-１）DIPスイッチによって設定されたモードは、電源投入時または、リセットボタンを押したときに有効となります。

画面モード	WIDTH文	DIP スイッチ
40文字×20行	WIDTH　40,20	
40文字×25行	WIDTH　40,25	
80文字×20行	WIDTH　80,20	
80文字×25行	WIDTH　80,25	

（図 3-1-1）

3-1-2 WIDTH文のパラメータの省略

WIDTH文は、桁数と行数の２つのパラメータをもちます。N-BASICでのWIDTH文はいずれも省略できますが（例えばWIDTH,25、WIDTH,）、N_{88}-BASICでは、行数の省略しかできません。したがって、桁数がわかっていないときに行数だけを変えたい場合は次のようにします。

WIDTH　PEEK（&HEF89），25（または20）

このEF89Hというのは画面の桁数がはいっているアドレスです。ちなみに行数はその前のEF88H番地にはいっています。

ただしこれは、BASICインタプリタが画面モードの値を参照するためのものであって、これらのアドレスに値をPOKEしたからといって画面モードは変化しません（他にCRTCのモード設定が必要です）。

3-1-3 WIDTH文とスクロールウィンドウ

N_{88}-BASICにおけるWIDTH文は、N-BASICのものといくらか相違点があります。前に述べたパラメータの省略の他に次のようなところが異なっています(図3-1-3)。

N_{88}-BASIC	N-BASIC
WIDTH 文を実行すると必ず画面がクリアされる	桁数を変化させた場合にのみ画面がクリアされる
WIDTH 文の実行によりスクロールウィンドウが初期化される（CONSOLE 0, 25 が行われる）	スクロールウィンドウは変化しない。

(図3-1-3)

第1の点については、困った点もでてきます。N-BASICでは、DMACやCRTCのモードの再設定（DMAをOFFにした場合のリカバー、OUT 81,33による画面反転をもとに戻すなど）のために、WIDTH, を実行すればよかったわけですが、N_{88}-BASICではそれを行うと消えては困る画面が消されてしまうことになります。

第2の点については、WIDTH文を実行することでスクロールウィンドウが初期化できるという利点がありますが、そうであって欲しくないときもあります。

これについては、WIDTH文を実行する前にスクロールウィンドウを覚えておいて実行後に再びCONSOLE文を実行すれば良いわけです。

プログラム例を示しておきます。

```
1000 STLN=PEEK(&HE6B2)-1
1010 SCLN=PEEK(&HE6B3)-STLN
1020 WIDTH 80,25
1030 CONSOLE STLN,SCLN
1040 PRINT CHR$(11);
```

3-1-4 画面とVRAMアドレスの対応

テキスト画面に表示される文字は、VRAM（Video RAM）と呼ばれるN_{88}-BASICワークエリア上のデータです。画面の1文字は、VRAMの1バイトに対応し、このVRAMにデータを書き込むことにより、画面に文字を表示することができます。また、このVRAMのデータを読むことで画面に表示されている文字コードを知ることもできます。

このVRAMには、各行について120バイト（80バイトが文字コード、40バイトがアトリビュートコード）ずつ、合計3000バイトが使われています。

実際のVRAMのアドレスは次の表の通りです。

VRAMメモリマップ

Line	Character	Attribute
0	F3C8...F417	F418...F43F
1	F440...F48F	F490...F4B7
2	F4B8...F507	F508...F52F
3	F530...F57F	F580...F5A7
4	F5A8...F5F7	F5F8...F61F
5	F620...F66F	F670...F697
6	F698...F6E7	F6E8...F70F
7	F710...F75F	F760...F787
8	F788...F7D7	F7D8...F7FF
9	F800...F84F	F850...F877
10	F878...F8C7	F8C8...F8EF
11	F8F0...F93F	F940...F967
12	F968...F9B7	F9B8...F9DF
13	F9E0...FA2F	FA30...FA57
14	FA58...FAA7	FAA8...FACF
15	FAD0...FB1F	FB20...FB47
16	FB48...FB97	FB98...FBBF
17	FBC0...FC0F	FC10...FC37
18	FC38...FC87	FC88...FCAF
19	FCB0...FCFF	FD00...FD27
20	FD28...FD77	FD78...FD9F
21	FDA0...FDEF	FDF0...FE17
22	FE18...FE67	FE68...FE8F
23	FE90...FEDF	FEE0...FF07
24	FF08...FF57	FF58...FF7F

次に、カーソル位置からVRAMのアドレスを求める方法を示します。

※BASICによる方法

• ４０ケタモード

V.ADRS＝&HF3C8＋120＊CUR.Y＋2＊CUR.X

• ８０ケタモード

V.ADRS＝&HF3C8＋120＊CUR.Y＋CUR.X

これらは、関数として定義しておくと便利です。

※機械語による場合

Hレジスタに（桁＋１）、Ｌレジスタに（行＋１）を入れて、429DH番地をコールすると、HLレジスタペアにVRAMのアドレスが得られます。

例）５行、20桁のVRAMアドレスを求める。

```
            桁 行
LD     HL, 1506H
CALL   429DH
```

3-1-5 VRAM位置の移動

VRAMの位置は、N-BASICでは、F300H番地からに固定されていましたが、N_{88}-BASICでは、ポインタVRAMAD（E6C4，C5）で与えられます。

つまり、この値を変えることにより、VRAMの位置を別のところへ移すことができるわけです。

それでは実際にやってみましょう。

まず3000バイト分のRAMエリアを確保します。

```
CLEAR , &HCFFF
```

次にポインタを書き換えます。

```
POKE  &HE6C5 , &HD0 : POKE  &HE6C4,0
```

これで移ったのですが、キーを押してもカーソルが動くだけで画面はもとのままですね。これは、DMACがまだ前のままの状態であるためです。これを再セットするには、WIDTH文を実行します（一応CLRキーを押した後で行って下さい）。

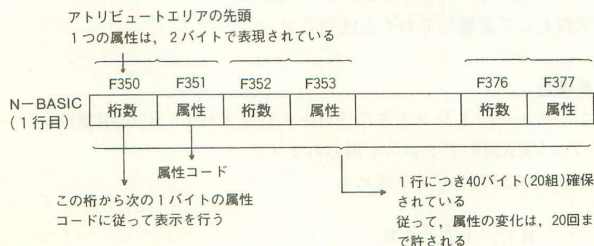
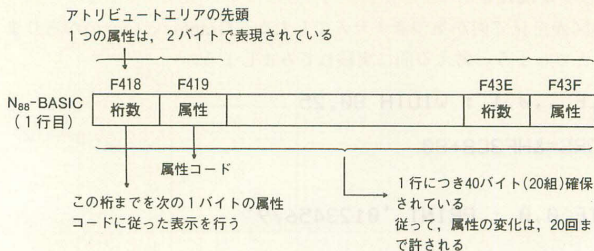
これで必要な操作は終わりです。

```
POKE  &HD000 , ASC ("A")
```

を実行すると左上にAという文字が出るはずです。これでVRAMは、D000H番地からDBB7H番地に移りました。

3-2. アトリビュートエリア

PC-8801ではテキスト画面の制御にμPD-3301を使っています(PC-8001でも同じものを使っています)。これは、カラーやリバースなどの指定にアトリビュート（属性）方式を用いているものです。この属性は、前に示したようにVRAM上にアトリビュートエリアとして置かれ、次のような構成になっています。

N-BASICとN_{88}-BASICでは多少違いがありますが、本質的には同じものです。

3-2-1 属性コード

属性コードは、1バイトよりなり、白黒モード、カラーモードのそれぞれの場合について次のような意味をもちます。

(a)白黒モード（CONSOLE ，，，0）

7	6	5	4	3	2	1	0
Graphic=1 Character=0	0	アンダーライン	アッパーライン	0	リバース	ブリンク	シークレット

000	ノーマル	COLOR 0
001	シークレット	COLOR 1
010	ブリンク	COLOR 2
011	シークレット	COLOR 3
100	リバース	COLOR 4
101	リバースシークレット	COLOR 5
110	リバースブリンク	COLOR 6
111	リバースシークレット	COLOR 7

(b)カラーモード（CONSOLE ，，，1）

7	6	5	4	3	2	1	0
0	0	アンダーライン	アッパーライン	0	リバース	ブリンク	シークレット

※カラー時の LINE 文に対応

このビットが0か1か2通りある。

7	6	5	4	3	2	1	0
緑	赤	青	Graphic=1 Character=0	1	0	0	0

000	黒	COLOR 0
001	青	COLOR 1
010	赤	COLOR 2
011	紫(マゼンダ)	COLOR 3

100	緑	COLOR 4
101	水色(シアン)	COLOR 5
110	黄	COLOR 6
111	白	COLOR 7

3-2-2 グラフィックが使える

属性コードの図表を見て何か気づきませんでしたか。Graphicというのがありますね、これはどういうことでしょう。考える前に実験してみましょう。

```
100 CONSOLE ,,0,1 : WIDTH 80,25
110 '
120 ATR.ADRS=&HF3C8+80
130 '
140  CLS
150  LOCATE 0,0 : PRINT "012345679"
160 '
170  POKE ATR.ADRS  ,   5 : ' col.adrs
180  POKE ATR.ADRS+1,&HF8 : ' 11111000 ' color = 3 , graphic
190 '
200 END
```

画面左上の最初の5桁がグラフィック表示になりましたね。

実はN_{88}-BASICでもアトリビュートの操作によってN-BASICでの（160×100，80×100）ドットグラフィックが使えるのです（ハードウェアは同じものなので当然のことです）。

となれば、このLOW RES（ロウ・レゾルーション）グラフィックを放っておくのももったいないですね。

でもN_{88}-BASICではLOW RESのPSET，PRESET文、ないしはアトリビュートの操作もBASICでできるほど簡単ではありません。

次の節ではこのアトリビュートのセットの仕方について考えてみましょう。

3-2-3 アトリビュート セット

アトリビュートのセットの中でも文字の色付けはBASICインタプリタで行なってくれます。ところがアンダーライン、アッパーライン、グラフィックモードなどについては自力でやるしかありません。そこでここでは、2つの方法でアトリビュートのセットを行った例を示します。

① 置きかえ法

この方法は色付けをBASICでやってくれるのでそれを利用するというものです。ここでは白黒モードで話を進めますがカラーモードでも同様のことができます。

原理としては、画面上では使わない色（たとえばCOLOR 1）を使って画面にデータを書いた後、アトリビュート・エリアをサーチしてそのアトリビュート・コードを目的のコードで置きかえるというものです。

例としてグラフィックによる絵を出すプログラムをあげておきます。

```
100 CONSOLE ,,,0 : WIDTH 80,25
110 '
120 DEF FNAT(LN)=&HF418+120*LN
130 '
140 CLS
150 '
160  PAT.A$=CHR$(&HC8)+CHR$(&HF6)+CHR$(&H6F)+CHR$(&H8C)
170  PAT.B$=CHR$(&H59)+CHR$(&H5B)+CHR$(&HB5)+CHR$(&H95)
180 '
190  X=35 : Y=11
200  COLOR 1
210  LOCATE X,Y   : PRINT PAT.A$
220  LOCATE X,Y+1 : PRINT PAT.B$
230  COLOR 0
240 '
250  SRCH.CODE=1    : ' secret character
260  SET.CODE=&H80 : ' graphic mode
270  ATR.ADRS=FNAT(Y)    : GOSUB *ATR.RESET
280  ATR.ADRS=FNAT(Y+1) : GOSUB *ATR.RESET
290 '
300 END
310 '
320 *ATR.RESET
330  FOR I=1 TO 39 STEP 2
340    IF PEEK(ATR.ADRS+I)=SRCH.CODE THEN POKE ATR.ADRS+I,SET.CODE
350  NEXT
360 RETURN
```

もう1つ、入力した文字列にアンダーラインを引くプログラムをあげておきましょう。

```
100 CONSOLE ,,,0 : WIDTH 80,25
110 '
120 DEF FNAT(LN)=&HF418+120*LN
130 '
140 CLS
150 '
160  PRINT "Input Strings ( max = 39 )"
170  LINE INPUT STRNG$
180 '
190  Y=10
200  COLOR 1
210  LOCATE 0,Y   : PRINT STRNG$
220  COLOR 0
230 '
240  SRCH.CODE=1    : ' secret character
250  SET.CODE=&H20 : ' graphic mode
260  ATR.ADRS=FNAT(Y)    : GOSUB *ATR.RESET
270 '
280 END
290 '
300 *ATR.RESET
310  FOR I=1 TO 39 STEP 2
320    IF PEEK(ATR.ADRS+I)=SRCH.CODE THEN POKE ATR.ADRS+I,SET.CODE
330  NEXT
340 RETURN
```

② ROM内ルーチンの利用

アトリビュートのセットにはROM内のルーチンも利用できます。使い方は、HLレジスタに画面上のアドレス、Cレジスタにアトリビュート・コードを入れて4351H番地をCALLするだけでです。

機械語ルーチンをCALLするにはUSR関数とCALL文があるわけですが、ここではDISK-BASICでなくても使えるようにUSR関数を使った例をあげておきます。残念ながらUSR関数では2つのパラメータを渡すことができませんので、360行にあるようにアトリビュート・コードはPOKE文で与えています。

```
100 '--- write atr.set M.L.
110  FOR I=&HF320 TO &HF329
120   READ DA$
130   POKE I,VAL("&H"+DA$)
140  NEXT
150  DEF USR=&HF320
160 '
170  DATA 0E,00,7E,23,66,6F,CD,51,43,C9
180 '
190 '--- test program
200 CONSOLE ,,,1 : WIDTH 80,25
210 DEFINT A-Z
220 '
230 CLS
240 '
250  FOR I=0 TO 29
260   CLR=INT(RND*7)+1
270   X=INT(RND*80)
280   Y=INT(RND*20)
290   V.ADRS=&HF3C8+Y*120+X
300   LOCATE X,Y : PRINT CHR$(1);
310   GOSUB *ATR.SET.SUB
320  NEXT
330 '
340 END
350 '
360 *ATR.SET.SUB
370  POKE &HF321,CLR*&H20+&H18
380  DUMMY=USR(V.ADRS)
390 RETURN
```

この例では、アトリビュート・コードを、グラフィックモードにするというものです。一種のLOW RES PSET文とでも考えて下さい。アトリビュート・コードは370行で決めています。

3-3. テキスト画面のGET、PUT

N-BASICでは、テキスト画面のGET（画面のデータを配列に取り込む）、PUT（配列のデータを画面に表示する）が使えましたが、N_{88}-BASICでは、グラフィック画面のGET、PUTしかできません。

ここでは、BASICと機械語を組み合わせてアトリビュートも含めたテキスト画面のGET、PUTサブルーチンを作ってみました。

これは、N-BASICでのGET@A、PUT@Aと同じようなもので、(GX1, GY1)－(GX2, GY2）の間の画面データを整数型配列ARYに読み込んだり、逆に、画面に表示するためのサブルーチンです。

・テキスト画面GET、PUTサブルーチン

```
1000 '----- GET@ SUB ROUTINE -----
1010 'GET@A(GX1,GY1)-(GX2,GY2),ARY
1020 '-----------------------------
1030 '
1040 *GET.SUB
1050  ERASE ARY
1060  GXS=GX2-GX1+1 : GYS=GY2-GY1+1
1070  DIM ARY(GXS*GYS)
1080  FOR Y=1 TO GYS
1090   FOR X=1 TO GXS
1100     VAD=&HF3C8+120*(GY1+Y-1)+(GX1+X-1)
1110     ARY(GXS*(Y-1)+X)=USR0(VAD)
1120   NEXT X
1130  NEXT Y
1140 RETURN
2000 '----- PUT@ SUB ROUTINE -----
2010 'PUT@A(GX1,GY1)-(GX2,GY2),ARY
2020 '-----------------------------
2030 '
2040 *PUT.SUB
2050  GXS=GX2-GX1+1 : GYS=GY2-GY1+1
2060  FOR Y=1 TO GYS
2070   FOR X=1 TO GXS
2080     POKE &HF2FE,GY1+Y
2090     POKE &HF2FF,GX1+X
2100     DM=USR1(ARY(GXS*(Y-1)+X))
2110   NEXT X
2120  NEXT Y
2130 RETURN
```

（1100行の注）40桁モードで使う場合は'*2'を付ける

使用する変数はすべて整数型にしておいて下さい。

配列ARYの1つの要素には1つの文字コードとアトリビュートコードがはいります。

ARY(X)	アトリビュートコード	文字コード
	1バイト	1バイト

次に簡単なテストプログラムをあげておきます

```
100 '-------------------------------
110 '    GET@A,PUT@A  SUBROUTINE
120 '-------------------------------
130 '
140 '----- INIT -----
150 WIDTH 80,25 : CONSOLE ,,,1
160 CLS
170 DEFINT A-Z
180 DEF USR0=&HF2F0
190 DEF USR1=&HF2E0
200 DIM ARY(0)
210 FOR I=&HF2E0 TO &HF2FC
220   READ D$ : POKE I,VAL("&H"+D$)
230 NEXT
240 DATA 46,23,4E,C5,2A,FE,F2,CD,9D,42,C1,CD,50,43,C9,00
250 DATA E5,7E,23,66,6F,CD,52,44,E1,77,23,71,C9
260  ----- TEST PROGRAM -----
270  GX1=0 : GY1=0
280  GX2=4 : GY2=2
290  COLOR 4 : LOCATE GX1  ,GY1   : PRINT "┌───┐"
300  COLOR 2 : LOCATE GX1+2,GY1   : PRINT "♥";
310  COLOR 4 : LOCATE GX1  ,GY1+1 : PRINT "│   │"
320  COLOR 5 : LOCATE GX1+1,GY1+1 : PRINT "*┼*";
330  COLOR 4 : LOCATE GX1  ,GY1+2 : PRINT "└───┘"
340  GOSUB *GET.SUB
350  FOR I=1 TO 20
360    GX1=RND*70 : GY1=RND*20
370    GX2=GX1+4 : GY2=GY1+2
380    GOSUB *PUT.SUB
390  NEXT
400 END
1000 '----- GET@ SUB ROUTINE -----
1010 'GET@A(GX1,GY1)-(GX2,GY2),ARY
1020 '----------------------------
1030 '
1040 *GET.SUB
1050  ERASE ARY
1060  GXS=GX2-GX1+1 : GYS=GY2-GY1+1
1070  DIM ARY(GXS*GYS)
1080  FOR Y=1 TO GYS
1090   FOR X=1 TO GXS
1100     VAD=&HF3C8+120*(GY1+Y-1)+(GX1+X-1)
1110     ARY(GXS*(Y-1)+X)=USR0(VAD)
1120   NEXT X
1130  NEXT Y
1140 RETURN
2000 '----- PUT@ SUB ROUTINE -----
2010 'PUT@A(GX1,GY1)-(GX2,GY2),ARY
2020 '----------------------------
2030 '
2040 *PUT.SUB
2050  GXS=GX2-GX1+1 : GYS=GY2-GY1+1
2060  FOR Y=1 TO GYS
2070   FOR X=1 TO GXS
2080     POKE &HF2FE,GY1+Y
2090     POKE &HF2FF,GX1+X
2100     DM=USR1(ARY(GXS*(Y-1)+X))
2110   NEXT X
2120  NEXT Y
2130 RETURN
```

3-4. PRINT文テクニック

3-4-1 PRINT文と改行

リセット直後と、WIDTH文を実行した後で、次の文を実行してみて下さい。

PRINT STRING$(35,"#");"0123456789"

①リセット直後（40桁モード）

```
print string$(35,"#");"0123456789"
###################################01234
56789
Ok
```

②WIDTH40を実行後

```
print string$(35,"#");"0123456789"
###################################
0123456789
Ok
```

マニュアルの通り※なら②のようになるはずですが、リセットした後WIDTH文を実行しないと、①のように改行されずに続けて表示されることになります。

（※PRINT文で表示する文字列の長さ、数値の長さが、現在のカーソルのある位置より後方にとれない場合、改行して表示される。）

これはワークエリア　E64FH番地の値によるもので、一種のフラグとして考えてもかまいません。

リセットの時、ここにはFFHが書き込まれます。（これは②のような機能を無視するということです）また、WIDTH文を実行すると、その桁数が書き込まれます。つまり、文字列等を表示する場合、カーソルの位置がその値を超えるようであれば改行して表示するということを示しています。

ですから、N-BASICのように、①のような表示をしたければ、WIDTH文を実行した後であっても、E64FH番地に、FFHをPOKEしてやればよいわけです。

これを使って次のようなこともできます。このプログラムは、80桁モードで平方根の値を表示するのに、画面の左半分だけに出力するものです。あたかも、WIDTH PRINT（?）を実行したかのように動作していますね。

```
100 WIDTH 80,25
110 POKE &HE64F,40
120 PRINT "0....5....0....5....0....5....0....5....0"
130 '
140 FOR I=0 TO 50
150   PRINT SQR(I);
160 NEXT
```

```
0....5....0....5....0....5....0....5....0
 0  1  1.41421  1.73205  2  2.23607
 2.44949  2.64575  2.82843  3  3.16228
 3.31662  3.4641  3.60555  3.74166
 3.87298  4  4.12311  4.24264  4.3589
 4.47214  4.58258  4.69042  4.79583
 4.89898  5  5.09902  5.19615  5.2915
 5.38517  5.47723  5.56777  5.65686
 5.74456  5.83095  5.91608  6  6.08276
 6.16441  6.245  6.32456  6.40312
 6.48074  6.55744  6.63325  6.70821
 6.78233  6.85566  6.9282  7  7.07107
Ok
```

3-4-2 PRINT文とTAB関数

TAB関数の働きが、N-BASICとN_{88}-BASICでは異なります。N-BASICのプログラムをN_{88}-BASICのプログラムに直すときに注意したいことの1つです。

①値が表示桁数をこえる場合

（N_{88}-BASIC）

値を表示桁数で割った余りが、値となります。（ただし、WIDTH文を少なくとも1回実行しておく必要があります）。

```
print "A"tab(43)"B"
A  B
Ok
```

（40桁モード）

（N-BASIC）

表示桁数に関係なく値の数だけ空白を出力します。

```
print "A"tab(43)"B"
A
   B
Ok
```

（40桁モード）

②プリント位置がTABの値を起えている場合。

（N_{88}-BASIC）

改行して次の行のTABの位置まで空白を出力します。

```
print "0123456789"tab(7)"**"tab(2)"=="
0123456789
       **
  ==
Ok
```

（40桁モード）

（N-BASIC）

TABは無効となります。

```
print "0123456789"tab(7)"**"tab(2)"=="
0123456789**==
Ok
```

（40桁モード）

この他にも、N_{88}-BASICでは、TABの値は−32768〜32767までとれますが、N-BASICでは、0〜255しかとれない、という違いもあります。

3-4-3 PRINT文で矢印を書く

キャラクタコード表を見ると、'↑'とか'C_L'などの直接PRINT文では書けない文字がありますね。特に矢印は使ってみたいものの1つです。PRINT文がだめなら、直接VRAMにPOKEしてしまうのも1つの手ではありますが、VRAMの位置を計算したり、色を付けたりするのがどうも面倒です。

そこで、これらの文字を出力させるテクニックを紹介しましょう。

```
POKE &HE6B6,1
```

を実行してみて下さい。

OKのあとにC_R L_Fというように出ましたね。カーソルを移動させると矢印が出ます。

もとに戻したければ、次のようにして下さい。

```
POKE &HE6B6,0
```

これはプログラム中でも使うことができます。次のプログラムで確かめてみましょう。

```
100 WIDTH 40,25
110 POKE &HE6B6,1
120  LOCATE 17,10
130  PRINT CHR$(30);
140  LOCATE 16,11
150  PRINT CHR$(29)"□"CHR$(28);
160  LOCATE 17,12
170  PRINT CHR$(31);
180 POKE &HE6B6,0
190 END
```

PRINT文の後に"；"がついていますが、これは、'C_R L_F'という文字が出てしまうのを避けるためです。

第4章　グラフィック画面

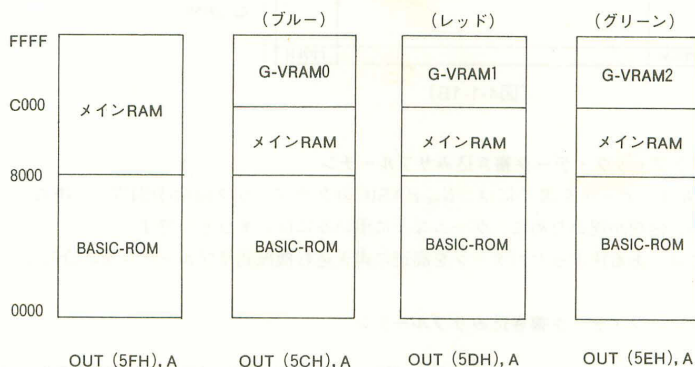
第4章　グラフィック画面

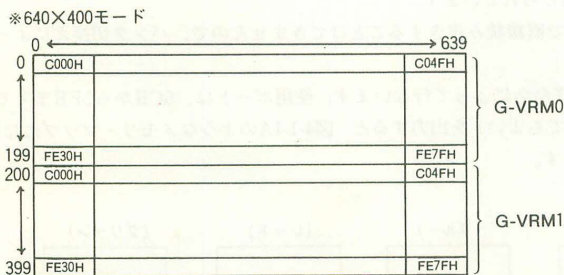
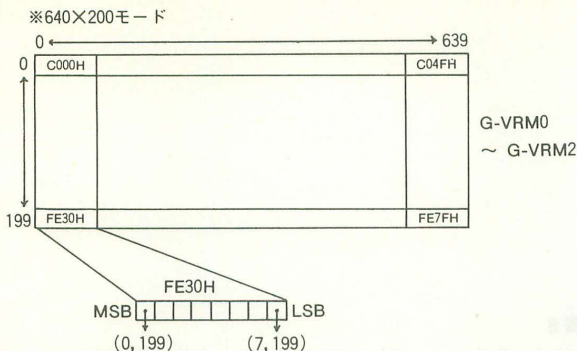
4-1. G-VRAM

4-1-1 G-VRAMの読み書き

グラフィックVRAMは、メモリ・マップ上のC000H番地からFFFFH番地までの16Kバイト×3バンクに割り当てられています。

G-VRAMは、BASICで直接読み書きすることはできませんので、バンク切換えによって機械語でアクセスします。

バンク切換えは、OUT命令によって行ないます。使用ポートは、5CHから5FHまでで、各ポートに、データ（何でもよい）を出力すると、図4-1-1Aのようなメモリ・マップになり、アクセスが可能になります。

（図4-1-1A）

グラフィック画面とG-VRAMとの対応は次の通りです。

(図4-1-1B)

4-1-2 グラフィック・データ書き込みサブルーチン

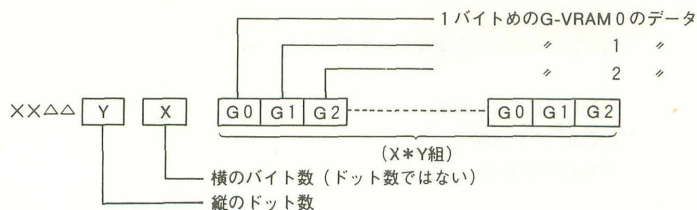
G-VRAMにデータを書くには、N88-BASICのグラフィック命令(PSET,LINEなど)を使いますが、処理が遅いために、ゲームなどに用いるにはいまひとつです。

ここでは、ある決まったパターンを高速に書き込む機械語サブルーチンを紹介します。

・グラフィック・データ書き込みサブルーチン

```
866A 00 7E 23 66 6F ED 5B 60 F2 EB 4E 23 46 23 EB C5
867A F3 1A D3 5C 77 D3 5F 13 1A D3 5D 77 D3 5F 13 1A
868A D3 5E 77 D3 5F 13 FB 23 10 E6 C1 C5 3E 50 90 4F
869A 06 00 09 C1 0D 20 D8 C9
```

このサブルーチンはリロケータブルになっていますのでRAM上のC000H番地より前ならどこにでも置くことができます。上の例は、N88-DISK-BASICの場合で、モニタのコマンドメッセージの部分を使いました。N88-ROM-BASICでは、ファイルバッファ#0のところにでも置くとよいでしょう。

• データの形式

F260 △△ ×× データの格納されている先頭番地

例）横16ドット，縦4ドットのパターンのデータをE000Hから入れる場合

16ドット / 4ドット

Data1	Data2
Data3	Data4
Data5	Data6
Data7	Data8

E000 04 02 ×× ×× ×× ×× ×× ×× …… ×× ×× ××
(Data1) (Data2) (Data8)

F260 00 E0

• 使い方

グラフィック画面はカラーモードにしておいて下さい。(SCREEN 0)

　DEF USR＝&H866A （機械語サブルーチンの先頭）

でUSR関数を定義しておきます。

あとは、G-VRAMのアドレスを引数としてUSR関数を使えば、データが表示されます。

　DUMMY＝USR(&HC000)

引数は整数型でなければなりません。

それでは実際に使ってみましょう。

上のサブルーチンが正しく入力されていることを確認して、次のデータを入力します。

(CLEAR,&HDFFFを行っておくこと)

```
E000 10 04 00 FF FF 00 FF FF 00 FF FF 00 F0 F0 00 FF
E010 FF 00 FF FF 00 FF FF 00 F0 F0 00 FF FF 00 FF FF
E020 00 FF FF 00 F0 F0 00 FF FF 00 FF FF 00 FF FF 00
E030 F0 F0 00 86 86 00 0C 0C 00 18 18 00 30 30 01 79
E040 01 F0 F3 F0 07 E7 07 C0 C0 C0 07 F7 07 E0 EF E0
E050 1F DF 1F 80 B0 80 0F EF 0F C0 DF C0 3F BF 3F 00
E060 70 00 1F DF 1F 80 BF 80 7E 7E 7E 00 F0 00 3F BF
E070 3F 00 7E 00 FC FD FC 00 F0 00 7C 7C 7C 01 F9 01
E080 F0 F3 F0 00 E0 00 00 83 83 00 06 06 00 0C 0C 00
E090 10 10 00 FF FF 00 FF FF 00 FF FF 00 F0 F0 00 FF
E0A0 FF 00 FF FF 00 FF FF 00 F0 F0 00 FF FF 00 FF FF
E0B0 00 FF FF 00 F0 F0 00 FF FF 00 FF FF 00 FF FF 00
E0C0 F0 F0

F260 00 E0
```

DEF USR=&H866A

DUMMY=USR(&HC000)

を実行すると、画面左上になにやら変なマークが出ましたね。

機械語で使うときは次のようにします。

HL=G-VRAMのアドレス

DE=データ格納アドレス

を設定した後

CALL　8673H

このサブルーチンはカラーモード用ですので、白黒モードで使う場合には、不要なバンクのデータを0にしておくとよいでしょう。

機械語に自信のある方は、ソース・リスト（付-1）を参考に白黒モード専用のサブルーチンを作ってみて下さい。

4-1-3 グラフィック・データ・ジェネレータ

4-1-2のグラフィック・データ書き込みサブルーチンはどうでしたか。高速なのはよいのですが、データを作るのがめんどうですね。

そこで、このデータを作成するプログラムを作ってみました。先程のデータも、このプログラムを使って作成したものです。

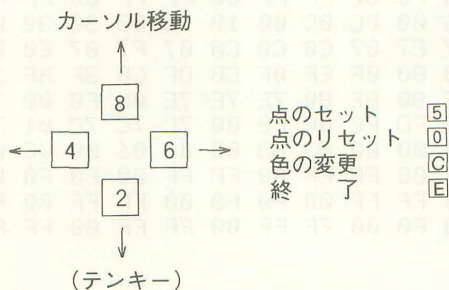
このプログラムでは、G-VRAMのデータを読むのに、1-6の拡張PEEK文を使っています。

810行でエラーが出る場合は、1-6のプログラムを実行した後、このプログラムを実行して下さい。

まず、作成したいパターンの大きさを入力します。(横48、縦24が最大です)。

あとは、カーソルをテンキーで操作してパターンを作ります。(図4-1-3)

終ったら、しばらくたってデータが入っているアドレスが表示されますので、必要ならば、その範囲のデータをセーブしておいて下さい。

N_{88}-DISK-BASICの場合は、920行の先頭の ′ を除くと自動的にセーブされます。

(図4-1-3)

```
100 '
110 '  Graphic Data Generator
120 '
130 CLEAR ,&HDFFF
140 CONSOLE 0,25,0,1 : WIDTH 80,25
150 COLOR 7
160 SCREEN 0,3 : CLS 3 : SCREEN 0,0
170 DEFINT A-Z
180 '
190  INPUT "INPUT MAX.X(dot),MAX.Y(dot)";MAX.X,MAX.Y
200  IF MAX.X<1 OR MAX.X>48 THEN 190
210  IF MAX.Y<1 OR MAX.Y>24 THEN 190
220 '
230  MAX.BX=(MAX.X-1)¥8 : MAX.BY=MAX.Y-1
240 '
250  CLS
260  LOCATE 23,0
270  FOR X=1 TO MAX.X
280    PRINT HEX$(X MOD 10);
290  NEXT X
300  FOR Y=1 TO MAX.Y
310    LOCATE 20,Y : PRINT USING "##:";Y;
320  NEXT Y
330  LINE(182,7)-(183+MAX.X*8,8+MAX.Y*8),7,B
340  CUR.X=23 : CUR.Y=1
350  COL=7
360 ' --- main loop ---
370 *MAIN
380  LOCATE CUR.X,CUR.Y
390  IN$=INPUT$(1)
400  ON INSTR("246850",IN$) GOSUB *DW,*LF,*RT,*UP,*ST,*RS
410   IF IN$="E" OR IN$="e" THEN *GN.END
420   IF IN$="C" OR IN$="c" THEN GOSUB *CH.COLOR
430  GOTO *MAIN
440 ' --- Cursor move ---
450 *DW
460  IF CUR.Y<MAX.Y    THEN CUR.Y=CUR.Y+1
470 RETURN
480 *LF
490  IF CUR.X>23       THEN CUR.X=CUR.X-1
500 RETURN
510 *RT
520  IF CUR.X<MAX.X+22 THEN CUR.X=CUR.X+1
530 RETURN
540 *UP
550  IF CUR.Y>1        THEN CUR.Y=CUR.Y-1
560 RETURN
570 '
580 *ST
590  PRINT "●"; : PSET(CUR.X-23,CUR.Y-1),COL
600 RETURN
```

```
610 *RS
620  PRINT " "; : PSET(CUR.X-23,CUR.Y-1),0
630 RETURN
640 ' --- change color ---
650 *CH.COLOR
660  LOCATE 0,20 : PRINT "COLOR ?";
670  CL$=INPUT$(1)
680  IF CL$<"1" OR CL$>"7" THEN 670
690  LOCATE 0,20 : PRINT "        ";
700  COL=VAL(CL$)
710  COLOR COL
720 RETURN
730 ' --- Gen.End ---
740 *GN.END
750  LOCATE 0,20 : PRINT "GEN.END"
760  AD=&HE000
770  POKE AD,MAX.BY+1 : AD=AD+1
780  POKE AD,MAX.BX+1 : AD=AD+1
790  FOR Y=0 TO MAX.BY
800    FOR X=0 TO MAX.BX
810      GV(0)=PEEK(&HC000+Y*80+X,"0")
820      GV(1)=PEEK(&HC000+Y*80+X,"1")
830      GV(2)=PEEK(&HC000+Y*80+X,"2")
840      FOR I=0 TO 2
850        POKE AD,GV(I): AD=AD+1
860      NEXT I
870    NEXT X
880  NEXT Y
890  CLS
900  SCREEN ,3
910  LOCATE 5,10 : PRINT "Used &HE000 - &H"HEX$(AD-1)
920 'BSAVE "GRPDAT.bin",&HE000,AD-&HE000
930 END
```

4-1-4 高速画面クリア

この節の最後として高速画面クリアコマンドを付加するプログラムを紹介します。

(N88-ROM BASIC 版)

```
100 '
110 '   High Speed CRT CLS
120 '
130 DEFINT A-Z
140 SA=VARPTR(#0)+9
150 '
160 FOR I=SA TO SA+30
170 READ D$ : POKE I,VAL("&H"+D$)
180 NEXT I
190 '
200 CH=&HEEB6:POKE CH,&HC3
210 POKE CH+1,PEEK(VARPTR(SA))
220 POKE CH+2,PEEK(VARPTR(SA)+1)
230 '
240 DATA 00,F3,D9,3E,5C,4F,ED,79,21,00,C0,11,01,C0,01,7F
250 DATA 3E,36,00,ED,B0,3C,FE,5F,20,EB,D3,5F,D9,FB,C9
```

(N88-DISK BASIC 版)

```
100 '
110 '   High Speed CRT CLS
120 '
130 FOR I=&H866A TO &H8688
140   READ D$ : POKE I,VAL("&H"+D$)
150 NEXT I
160 '
170 CH=&HEEB6:POKE CH,&HC3:POKE CH+1,&H6B:POKE CH+2,&H86
180 '
190 DATA 00,F3,D9,3E,5C,4F,ED,79,21,00,C0,11,01,C0,01,7F
200 DATA 3E,36,00,ED,B0,3C,FE,5F,20,EB,D3,5F,D9,FB,C9
```

上のプログラムを実行した後は、CMDを実行すると高速で画面がクリアされるようになります。ただし、VIEWポートの値は無視され、すべてのG-VRAMがクリアされますので、その点は注意して下さい。

4-2. カラーパレット

N88-BASICでは、グラフィック画面の色の指定のためにカラーパレットという考え方を用いています。

これは、画面に表示されている文字（漢字などのようにグラフィック画面に書かれているもの）や絵の色を瞬時にして変えることができるというすばらしい機能をもつものです。うまく利用すればグラフィック画面の操作の遅さを充分カバーできるものではないでしょうか。

そこでこの節では、カラーパレットの応用、機械語での制御の方法などについて説明します。

4-2-1 BASICによるカラーパレットの指定

カラーパレットの指定には

COLOR＝（〈パレット番号〉，〈カラーコード〉）

というステートメントを使います。

例えば

COLOR＝（2，4）

とすれば、パレット番号2がカラーコード4（すなわち緑）になるということです。

この操作は、非常に高速で、(OUT命令一発ですので当然ではありますが)、一瞬のうちに画面の色を変化させることができます。

ただ残念なことに画面との同期をとっていないためか、次のプログラムのように同じパレット番号を次々と変化させると、画面がチラついてしまいます。

```
100 SCREEN 0,0 : CLS 3
110 CIRCLE(320,100),150,7
120 PAINT(320,100),4,7
130 FOR I=1 TO 7
140    COLOR=(4,I)
150 NEXT
160 GOTO 130
```

これも､使い方によっては面白い効果が得られ､ゲームなどへの応用にはもってこいでしょう。

カラーパレットの指定のしかたは、前に述べましたが、このステートメントにはマニュアルにない書式があります。実は、「＝」の次の（　　）は何回もくり返して書けるのです。つまり、

COLOR＝（1，1），（2，2），（3，3）

というようなことができるわけです。

まあ、あまりたいした発見でもないのですが、こういうものは他にもあるようです。偶然を期待していろいろやってみるもよいし、ROMのインタプリタを解析して隠れコマンド、ステートメントを見つけるというのもパソコンユーザーにとっての楽しみではないでしょうか。

カラーパレット指定の高速性を利用して、簡単なアニメーションの応用も考えられます。次にこの応用例を紹介しておきましょう。

(例：回転する車輪)

```
100 SCREEN 0,0
110 CLS 3
120 CIRCLE(320,100),180,7
130 PI2.6=3.14159/3
140 PI2.36=3.14159/18
150 FOR I=1 TO 6
160   FOR J=1 TO 6
170     PX=COS(PI2.6*J+PI2.36*I)*180
180     PY=SIN(PI2.6*J+PI2.36*I)*90
190     LINE(320,100)-(320+PX,100+PY),I
200   NEXT
210 NEXT
220 FOR I=1 TO 6
230   COLOR=(I,0)
240 NEXT
250 FOR I=1 TO 6
260   COLOR=(I,7)
270   COLOR=(((I+4)MOD 6)+1,0)
280   FOR J=0 TO 20 : NEXT
290 NEXT
300 GOTO 250
```

4-2-2 機械語によるカラーパレットの制御

先程、カラーパレットの制御はOUT命令一発だと言いましたがそのとおりでOUT命令だけでこのカラーパレットは制御できます。

I/Oポートは、54H～5BHが割り当てられ次の表（4-2-2A）のように対応しています。

OUTPUT アドレス	パレット番号
5 4 H (01010100)	0
5 5 H (01010101)	1
5 6 H (01010110)	2
5 7 H (01010111)	3
5 8 H (01011000)	4
5 9 H (01011001)	5
5 A H (01011010)	6
5 B H (01011011)	7

(表4-2-2A)

パレット番号に対するカラーコードの指定は上の各ポートに次のデータを出力すればよいわけです。

データ	カラーコード
MSB 7 6 5 4 3 2 1 0 × × × × × G R B	
0 0 0	0（黒）
0 0 1	1（青）
0 1 0	2（赤）
0 1 1	3（紫）
1 0 0	4（緑）
1 0 1	5（水色）
1 1 0	6（黄）
1 1 1	7（白）

(表4-2-2B) ※上位 5 bitは無視されます。

OUT命令はBASICでは、

OUT &H56,3 ⇨ COLOR=(2, 3)と同じ

機械語では、

```
3E 03 LD  A,3
D3 56 OUT (56H),A
```

とします。

4-2-3 カラーパレットの初期化

カラーパレットは使って便利なものではあるのですが、1つのプログラムでカラーパレットを操作した後、別のプログラム(カラーパレットを使うことを前提としていない)で、思ったとおりの色がでなかったりするということがあります。これはカラーパレットを初期化していないためでカラーパレットを操作するプログラムや前に操作されたと思う場合は、原則として初期化(すべてのパレット番号と、カラーコードを同じにしておく)を行うように心がけましょう。

カラーパレットの初期化は、ダイレクトモードかプログラム中で

FOR I=0 TO 7:COLOR=(I,I):NEXT

とします。

入力するのがめんどうだという人には、ウォームスタート(STOPキーを押しながらリセットする)という手はどうでしょう。これなら画面も一気に消し去ることができて便利です。

機械語を使って作ることもできます。次に示すのは、いつでも使えるカラーパレットイニシャライズコマンド付加プログラムです。

これを1度実行しておけばリセットしない限り、CMDだけでカラーパレットの初期化ができるようになります。

```
100 ' ----- color palette initialize
110 FOR I=&HF320 TO &HF32B
120   READ DA$
130   POKE I,VAL("&H"+DA$)
140 NEXT
150 DATA 06,08,0E,5B,05,ED,41,04,0D,10,F9,C9
160 CMDHOK=&HEEB6
170 POKE CMDHOK,&HC3
180 POKE CMDHOK+1,&H20
190 POKE CMDHOK+2,&HF3
200 PRINT "Complete."
210 END
```

4-3. その他のグラフィック画面制御

4-3-1 バックグラウンドカラー

バックグラウンドカラーとは、グラフィック画面の地の色のことです。

N88-BASICで制御するには、COLOR文を用います。２番目のパラメータがバックグラウンドカラーになるわけですが、白黒モードと、カラーモードでは働きが違います。(マニュアルに記述されているのはカラーモードでのことです。) その違いを表にまとめておきます。

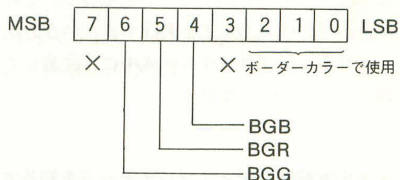
バックグラウンドカラーの違い

	白黒モード	カラーモード
パラメータの値	カラーコード	パレット番号
方　式	ハードウェアによる (COLOR文を実行すると瞬時にして色が変わる)	ソフトウェアによる (CLSやPRESETを実行すると作用した部分のみ色が変わる)

(表4-3-1)

白黒モードでのバックグラウンドカラーはI/Oポート52Hの3bitで制御されます。

カラーモードでのバックグラウンドカラーは、ソフトウェアによるものです。COLOR文の実行により、ワークエリアF01FH番地に、バックグラウンドカラーのパレット番号が書き込まれます。以後、N88-BASICでは、CLS文やPRESET文を実行するときにこの値を参照して、このパレット番号で、画面をクリアしたり、点を消したりします。

◦OUT PUT ポート　52H
◦データ

バックグラウンドカラー	BGG	BGR	BGB
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

(図4-3-1)

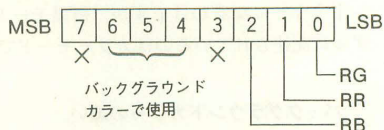
4-3-2 ボーダーカラー

ボーダーカラーは、画面のまわりの色のことで、COLOR文の3番目のパラメータで制御します。

但し、専用高解像度ディスプレイ(PC-8853など)を使用している時は、このパラメータの指定はできません(もし、指定するとFCエラーとなります)。

ボーダーカラーを機械語で制御するにはポート52Hの3bitを使います。

◦OUT PUTポート　52H

◦データ

ボーダーカラーで困ったことは、プログラム中で、COLOR文の3番目のパラメータを指定している場合です。普通のディスプレイでうまく動いたはずのプログラムが、専用高解像度ディスプレイにするとエラーとなってしまうことになります。対処法としては、ON ERROR GOTOで処理をするとか、次の方法でディスプレイモードを知って、それに合わせてCOLOR文を実行するという方法があります。プログラム作成中に、注意しておきたいことの一つです。

ボーダーカラー	RG	RR	RB
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

(図4-3-2)

・専用高解像度ディスプレイモードを知る方法

専用高解像度ディスプレイを使用する場合は、本体後部のジャンパースイッチをHにしますが、この状態は、IN命令によって知ることができます。

```
100 I40=INP(&H40)
110 IF I40 AND 2 THEN PRINT "Standard"; ELSE PRINT "High Scan";
120 PRINT " Display"
```

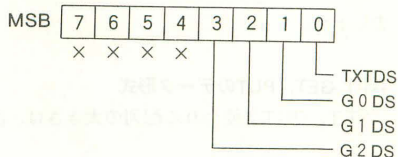
I40　AND　2＝2→標準ディスプレイ

I40　AND　2＝0→専用高解像度ディスプレイ

4-3-3 画面の重ね合わせ

PC-8801は、テキスト画面と、グラフィック画面の2種類の画面を持っています。

テキスト画面は、テキストVRAM、グラフィック画面は、グラフィックVRAMにデータを書き込むことによって、画面に文字や図形を表示することができるわけですが、CRT上にどの画面を表示するのかは、ハード的に制御することができます。

この画面制御のためのコントロールポートが53Hです。各bitの意味は次のようになります。

		0	1
TXTDS	（テキスト画面）	表示する	表示しない
G0DS	（G-VRM 0）	表示する	表示しない
G1DS	（G-VRM 1）	表示する	表示しない
G2DS	（G-VRM 2）	表示する	表示しない

（表4-3-3）

カラーグラフィックモードでは、TXTDSのみ意味を持ちます。つまり、このポートを操作しても、グラフィック画面は、表示されたままになるわけです。

N-BASICでは、テキスト画面を一時的に消しておきたいとき、DMAをとめるという方法を使っていました。N_{88}-BASICでもそれは可能なのですが、再表示のためにWIDTH文を実行すると、画面がクリアされて意味を持たなくなります。そこで、N_{88}-BASICではこのポートを使うことにします。

例えば、カラーグラフィックモードであれば、次のようにすればよいわけです。

OUT &H53,1　　…テキスト画面が消える
OUT &H53,0　　…テキスト画面が表示される。

ただし、DMAをストップする方法では、プログラムの実行速度が2～3割アップしますが、上記の方法では、実行速度は変化しません。

4-4. グラフィック画面のGET，PUT

N_{88}-BASICのGET,PUT文は、グラフィック画面に表示されているパターンを配列に読み込んだり、配列のデータをグラフィック画面に表示するための命令です。

ここでは、GET，PUTで使われるデータの形式や、PUT文の様々な表示方法を見ていきましょう。

4-4-1 GET，PUTのデータ形式

GET，PUTで使われる配列の大きさは、次の式で得られます。

〈添字の値〉＝〈必要なバイト数〉¥N＋〈OPTION BASE〉＋1—①
〈必要なバイト数〉＝〈横のバイト数〉＊〈縦のドット数〉＊M＋4—②

N：{ 整数型配列 N＝2 / 単精度型配列 N＝4 / 倍精度型配列 N＝8 }

M：{ 白黒モード M＝1 / カラーモード M＝3 }

式②の4という数字は、縦横のドット数をそれぞれ2バイトで表わすためのものです。

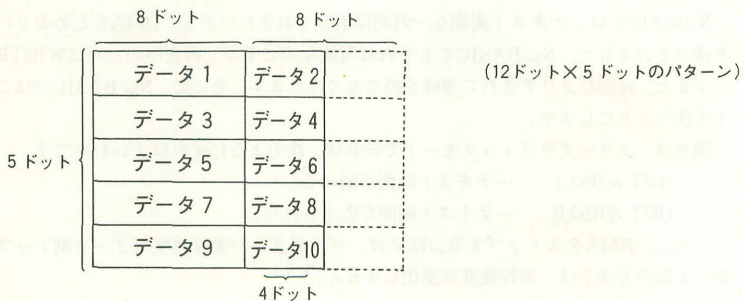
GET，PUTで使われる配列のデータは、次のような形になっています。

横のドット数	縦のドット数	データ1	データ2	データ3	データ4	・・・	…白黒モード
		データ1 B	データ1 R	データ1 G	データ2 B	・・・	…カラーモード

(12ドット×5ドットのパターン)

8ドット　8ドット

データ1	データ2	
データ3	データ4	
データ5	データ6	
データ7	データ8	
データ9	データ10	

5ドット　4ドット

まず最初に、4バイトのデータが、縦横のドット数を表わします。

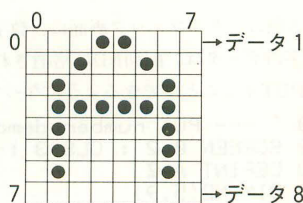
次に、グラフィックパターンのデータが、横8ドット、縦1ドットを1つの単位として白黒モードでは、1バイト、カラーモードでは3バイトのデータとなります。

したがって、上の例では、データのバイト数として、白黒モードでは、14バイト、カラーモードでは、34バイトが必要になるわけです。

また、このデータと配列の対応は、型や次元数によって次のようになります。

実際の例で見てみましょう。

右のようなパターンが、画面左上に表示されているとします。(640×200ドット，白黒モード)

これを整数型配列A%にGETします。

```
clear
Ok
dim a%(5)
Ok
get(0,0)-(7,7),a%
Ok
```

さて、ここで、配列A%の値をPRINTしてもよいのですが、テクノウ的に、配列が格納されているエリアを直接見てみましょう。

```
print hex$(varptr(a%(0)))
8621
Ok
mon
```

4-4-2 複数パターンを1つの配列に

N_{88}-BASICのGET，PUTでは、配列の要素を指定することで、複数のパターンを一つの配列に格納しておくことができます。

これは、異ったパターンを表示するときでも、要素の値を変数で持つことによって1つの文ですますことができるという利点があります。

ただし、このとき一次元の配列を用いると要素の計算や、大きさを決めるのがめんどうです。そこで、複数パターンのための2次元配列の使い方と、その応用例を紹介しましょう。

・配列宣言 (OPTION BASE 0)

DIM〈変数名〉(〈最も大きなパターンの添字の最大値〉，〈パターンの数〉－1)

・GET，PUTでの使い方

〈変数名〉(0,〈パターン番号〉)

※GET(0，0)－(7，7)，PAT(0，3)など

次の例は、グラフィック画面に、数字をPUTするサブルーチンです。

数字のデータは、配列NDに格納されており、NUMを与えることで、0から9までの数字をPUTすることができるようになっています。

```
100 ' --- PUT number demo ---
110 SCREEN 0,2 : CLS 3 : SCREEN 1,0
120 DEFINT A-Z
130 DIM ND(5,9)
140 FOR I=0 TO 9
150   FOR J=0 TO 5
160     READ DA$
170     ND(J,I)=VAL("&H"+DA$)
180   NEXT J
190 NEXT I
200 '
210 *NUM.DATA
220 DATA 8,8,423C,5A46,4262,3C
230 DATA 8,8,1808,0828,0808,3E
240 DATA 8,8,423C,0C02,4030,7E
250 DATA 8,8,423C,1C02,4202,3C
260 DATA 8,8,0C04,2414,047E,04
270 DATA 8,8,407E,0478,4402,38
280 DATA 8,8,201C,7C40,4242,3C
290 DATA 8,8,427E,0804,1010,10
300 DATA 8,8,423C,3C42,4242,3C
310 DATA 8,8,423C,3E42,0402,38
320 ' --- test ---
330 FOR NUM=0 TO 9
340   GX=NUM*60+40
350   GY=90
360   GOSUB *NUM.PUT
370 NEXT
380 END
390 ' --- number put sub ----
400 *NUM.PUT
410   PUT(GX,GY),ND(0,NUM)
420 RETURN
```

第5章　入出力ファイル

第5章　入出力ファイル

5-1. デバイス番号

BASICキーワードに中間言語があるように、デバイス名にもその内部表現として、1バイトのデバイス番号があります。

デバイス番号	デバイス名	デ バ イ ス
0～7	1：～8：	ディスク（ドライブ番号－1）
F8	SCRN：	スクリーン
F9	LPT1：	プリンタ
FA	CAS2：	カセット 600 ボー
FA	CAS1：	カセット 1200 ボー
FC	COM3：	RS232C チャネル3
FD	COM2：	〃 チャネル2
FE	COM1：	〃 チャネル1
FF	KYBD：	キーボード

表5－1

5-2. ファイル・ディスクリプタに変数が使える

ファイル・ディスクリプタは両端をダブルクォーテーションで囲って表わします。つまり文字列であるわけです。したがってファイル・ディスクリプタとして文字型の変数が使用できます。
もちろん、SAVE , LOADもできますが、LOADし終わると文字変数はクリアされてしまいます。

（例1）

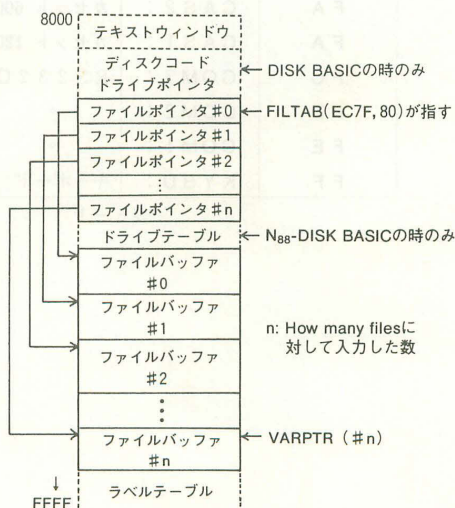
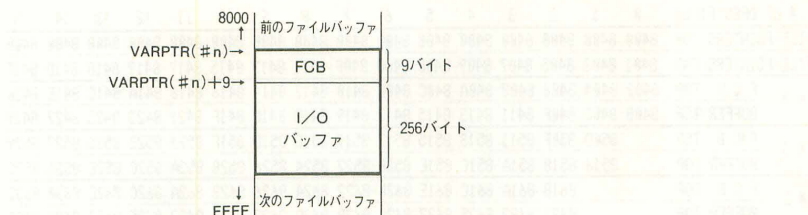
```
f$="FILE"
Ok
open f$ for output as #1
Ok
close
Ok
```

（例2）

```
f$="backup"
Ok
load f$+"n88"
Ok
print f$

Ok
```

5-3. ファイルバッファ

周辺装置とのデータのやりとりは原則としてファイルバッファを介して行なわれます。マニュアルでいう「窓口」にあたるわけです。

右図の様に＃0～＃nのファイルバッファがとられますが、各ファイルバッファの先頭アドレスを、ファイルポインタ（2バイト）が示しています。これはVARPTR（＃ファイル番号）で返されるものと同じです。

ファイルバッファは9バイトの作業領域（FCB）と256バイトのI/Oバッファから構成されています。

作業領域の内分けは次の図表の通りです。

FCB（ファイル　コントロール　ブロック）

VARPTR(♯n)	ファイルモード	bit 7 6 5 4 3 2 1 0 bit 0: FOR INPUT bit 1: FOR OUTPUT bit 2: ランダム アクセス bit 3: FOR APPEND bit 5: FOR IBM CLOSEされている時は00
＋1	先頭クラスタ番号	ファイルが格納されている先頭のクラスタ番号（ディスクファイルのみ）
＋2	現在のクラスタ番号	アクセス中のクラスタ番号（ディスクファイルのみ）
＋3	現在のセクタ番号	セクタ番号
＋4	デバイス番号	5－1参照
＋5	バッファ長	256バイトの時は00
＋6	データポインタ	シーケンシャルファイルの時、次に読み書きするデータの位置
＋7	ファイル属性	bit 7 6 5 4 3 2 1 0 bit 0: 機械語ファイル bit 4: プロテクト(P) bit 5: 特殊コード(E) bit 6: リードアフターライト(R) bit 7: 非アスキー形式 CLOSE されている時は00
＋8	仮想ヘッド位置	シーケンシャルファイルで、レコード（CRで区切られる文字列）中でのデータポインタの位置

N88-ROM BASIC

ファイルバッファアドレス一覧表

ファイル番号	# OF OPEN FILES	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	FILE POINTERS TOP	8400	8400	8400	8400	8400	8400	8400	8400	8400	8400	8400	8400	8400	8400	8400	8400
	FILE POINTERS END	8401	8403	8405	8407	8409	840B	840D	840F	8411	8413	8415	8417	8419	841B	841D	841F
# 0	F C B TOP	8402	8404	8406	8408	840A	840C	840E	8410	8412	8414	8416	8418	841A	841C	841E	8420
	BUFFER TOP	840B	840D	840F	8411	8413	8415	8417	8419	841B	841D	841F	8421	8423	8425	8427	8429
# 1	F C B TOP		850D	850F	8511	8513	8515	8517	8519	851B	851D	851F	8521	8523	8525	8527	8529
	BUFFER TOP		8516	8518	851A	851C	851E	8520	8522	8524	8526	8528	852A	852C	852E	8530	8532
# 2	F C B TOP			8618	861A	861C	861E	8620	8622	8624	8626	8628	862A	862C	862E	8630	8632
	BUFFER TOP			8621	8623	8625	8627	8629	862B	862D	862F	8631	8633	8635	8637	8639	863B
# 3	F C B TOP				8723	8725	8727	8729	872B	872D	872F	8731	8733	8735	8737	8739	873B
	BUFFER TOP				872C	872E	8730	8732	8734	8736	8738	873A	873C	873E	8740	8742	8744
# 4	F C B TOP					882E	8830	8832	8834	8836	8838	883A	883C	883E	8840	8842	8844
	BUFFER TOP					8837	8839	883B	883D	883F	8841	8843	8845	8847	8849	884B	884D
# 5	F C B TOP						8939	893B	893D	893F	8941	8943	8945	8947	8949	894B	894D
	BUFFER TOP						8942	8944	8946	8948	894A	894C	894E	8950	8952	8954	8956
# 6	F C B TOP							8A44	8A46	8A48	8A4A	8A4C	8A4E	8A50	8A52	8A54	8A56
	BUFFER TOP							8A4D	8A4F	8A51	8A53	8A55	8A57	8A59	8A5B	8A5D	8A5F
# 7	F C B TOP								8B4F	8B51	8B53	8B55	8B57	8B59	8B5B	8B5D	8B5F
	BUFFER TOP								8B58	8B5A	8B5C	8B5E	8B60	8B62	8B64	8B66	8B68
# 8	F C B TOP									8C5A	8C5C	8C5E	8C60	8C62	8C64	8C66	8C68
	BUFFER TOP									8C63	8C65	8C67	8C69	8C6B	8C6D	8C6F	8C71
# 9	F C B TOP										8D65	8D67	8D69	8D6B	8D6D	8D6F	8D71
	BUFFER TOP										8D6E	8D70	8D72	8D74	8D76	8D78	8D7A
#10	F C B TOP											8E70	8E72	8E74	8E76	8E78	8E7A
	BUFFER TOP											8E79	8E7B	8E7D	8E7F	8E81	8E83
#11	F C B TOP												8F7B	8F7D	8F7F	8F81	8F83
	BUFFER TOP												8F84	8F86	8F88	8F8A	8F8C
#12	F C B TOP													9086	9088	908A	908C
	BUFFER TOP													908F	9091	9093	9095
#13	F C B TOP														9191	9193	9195
	BUFFER TOP														919A	919C	919E
#14	F C B TOP															929C	929E
	BUFFER TOP															92A5	92A7
#15	F C B TOP																93A7
	BUFFER TOP																93B0

N88-DISK BASIC　　2ドライブ

ファイルバッファアドレス一覧表

ファイル番号

	# OF OPEN FILES	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	FILE POINTERS TOP	ADA6	ADA6	ADA6	ADA6	ADA6	ADA6	ADA6	ADA6	ADA6	ADA6	ADA6	ADA6	ADA6	ADA6	ADA6	ADA6
	FILE POINTERS END	ADA7	ADA9	ADAB	ADAD	ADAF	ADB1	ADB3	ADB5	ADB7	ADB9	ADBB	ADBD	ADBF	ADC1	ADC3	ADC5
# 0	F C B TOP	AEFC	AEFE	AF00	AF02	AF04	AF06	AF08	AF0A	AF0C	AF0E	AF10	AF12	AF14	AF16	AF18	AF1A
	BUFFER TOP	AF05	AF07	AF09	AF0B	AF0D	AF0F	AF11	AF13	AF15	AF17	AF19	AF1B	AF1D	AF1F	AF21	AF23
# 1	F C B TOP		B007	B009	B00B	B00D	B00F	B011	B013	B015	B017	B019	B01B	B01D	B01F	B021	B023
	BUFFER TOP		B010	B012	B014	B016	B018	B01A	B01C	B01E	B020	B022	B024	B026	B028	B02A	B02C
# 2	F C B TOP			B112	B114	B116	B118	B11A	B11C	B11E	B120	B122	B124	B126	B128	B12A	B12C
	BUFFER TOP			B11B	B11D	B11F	B121	B123	B125	B127	B129	B12B	B12D	B12F	B131	B133	B135
# 3	F C B TOP				B21D	B21F	B221	B223	B225	B227	B229	B22B	B22D	B22F	B231	B233	B235
	BUFFER TOP				B226	B228	B22A	B22C	B22E	B230	B232	B234	B236	B238	B23A	B23C	B23E
# 4	F C B TOP					B328	B32A	B32C	B32E	B330	B332	B334	B336	B338	B33A	B33C	B33E
	BUFFER TOP					B331	B333	B335	B337	B339	B33B	B33D	B33F	B341	B343	B345	B347
# 5	F C B TOP						B433	B435	B437	B439	B43B	B43D	B43F	B441	B443	B445	B447
	BUFFER TOP						B43C	B43E	B440	B442	B444	B446	B448	B44A	B44C	B44E	B450
# 6	F C B TOP							B53E	B540	B542	B544	B546	B548	B54A	B54C	B54E	B550
	BUFFER TOP							B547	B549	B54B	B54D	B54F	B551	B553	B555	B557	B559
# 7	F C B TOP								B649	B64B	B64D	B64F	B651	B653	B655	B657	B659
	BUFFER TOP								B652	B654	B656	B658	B65A	B65C	B65E	B660	B662
# 8	F C B TOP									B754	B756	B758	B75A	B75C	B75E	B760	B762
	BUFFER TOP									B75D	B75F	B761	B763	B765	B767	B769	B76B
# 9	F C B TOP										B85F	B861	B863	B865	B867	B869	B86B
	BUFFER TOP										B868	B86A	B86C	B86E	B870	B872	B874
#10	F C B TOP											B96A	B96C	B96E	B970	B972	B974
	BUFFER TOP											B973	B975	B977	B979	B97B	B97D
#11	F C B TOP												BA75	BA77	BA79	BA7B	BA7D
	BUFFER TOP												BA7E	BA80	BA82	BA84	BA86
#12	F C B TOP													BB80	BB82	BB84	BB86
	BUFFER TOP													BB89	BB8B	BB8D	BB8F
#13	F C B TOP														BC8B	BC8D	BC8F
	BUFFER TOP														BC94	BC96	BC98
#14	F C B TOP															BD96	BD98
	BUFFER TOP															BD9F	BDA1
#15	F C B TOP																BEA1
	BUFFER TOP																BEAA

[Apr 24, 1982] version

データの入出力はファイルバッファを介して行なわれると書きましたが、実際にI/Oバッファを使うのは、ディスクの入出力と、RS-232Cの入力の場合だけです。ディスク以外の出力はバッファを用いずに行なわれます。カセットからの入力はF0D3～F152Hのカセット用バッファを用い、キーボードからの入力はEFD9～EFF8Hのキー入力用バッファを用いています。

では実際にファイルバッファにどのように書き込まれていくか見てみましょう。DISKシステム、How many files＝2、使用するファイル番号＝#1とします。このとき前ページの表よりFCB：B009～B011H，I/Oバッファ：B012～B111Hとなります。
まずOPENします。

```
open "FILE" for output as#1
Ok
mon
hJdb009,b011
B009 02 75 75 00 00 00 00 02 00
hJdb012
B012 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
hJ^b
Ok
```

先頭クラスタ#（ディスケットによって異なります）
デバイス#（ドライブ1）
データポインタ（まだ何も書かれていない）
バッファ長＝256
属性なし
I/Oバッファはすべて00

次にデータを書き込みます。

```
print #1,"ABCDEF";
Ok
mon
hJdb009,b011
B009 02 75 75 01 00 00 06 08 06
hJdb012
B012 41 42 43 44 45 46 00 00 00 00 00 00 00 00 00 00
      A  B  C  D  E  F
hJ^b  → レコード
Ok
```

データポインタ（6文字書き込んだ）
まだCRが現れない

```
print #1,"XYZ"
Ok
mon

hJdb009,b011
B009 02 75 75 01 00 00 0B 08 01
hJdb012
B012 41 42 43 44 45 46 58 59 5A 0D 0A 00 00 00 00 00
     A  B  C  D  E  F  X  Y  Z  CR LF
hJ^b
Ok
```


長い文字列を書き込みます。

```
print #1,string$(250,"@")
Ok
mon

hJdb009,b011
B009 02 75 75 02 00 00 07 08 01
hJdb012
B012 40 40 40 40 40 0D 0A 00 00 00 00 00 00 00 00 00
     @  @  @  @  @  CR LF
hJ^b
Ok
```


CLOSEします。

```
close
Ok
mon

hJdb009,b011
B009 00 75 75 02 00 00 07 00 01
hJdb012
B012 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
hJ^b
Ok
```

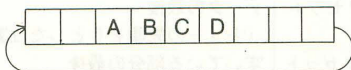
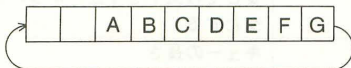
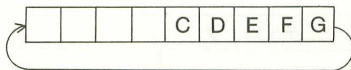
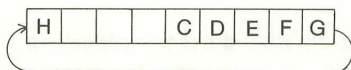
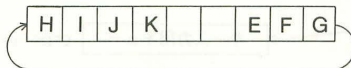
今の例ではクラスタ117に書き込みました。クラスタ117はPC-8031-2Wの場合はトラック29（1DH）,サーフェス0,セクタ9～16ですからモニタの^dコマンドで見てみます。

```
mon

h]^d1,0,1d,9
Track 1D,Surface 00,Sector 09
0000 41 42 43 44 45 46 58 59 5A 0D 0A 40 40 40 40 40
0010 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
0020 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
0030 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
0040 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
0050 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
0060 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
0070 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
0080 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
0090 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
00A0 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
00B0 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
00C0 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
00D0 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
00E0 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
00F0 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40

h]^d1,0,1d,a
Track 1D,Surface 00,Sector 0A
0000 40 40 40 40 40 0D 0A 1A 00 00 00 00 00 00 00 00
0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

データの終わりを示します。

5-4. キュー

前節で、キー入力とカセット入力では専用のバッファを使うと述べましたが、このバッファはキューと言われるものの仲間です。これは最初に入れたデータを最初にとり出すもので、FIFO（First In First Out）とも呼ばれます。

下図のようなキューがあるとします。データとして、A,B,C,Dが入っています。

データE,F,Gを入れます。

データA,Bを取り出します。

ここでデータHを入れるとこのようになります。

データC,Dをとり出し、データI,J,Kを入れます。

このようにしてデータの出し入れが行なわれますが、コンピュータがこの動作を行なうためにはいくつかの作業領域が必要です。

N_{88}-BASICでは次のものを持っています。

• 空いている部分の最後（データの先頭－1）の位置
• データの最後（空き部分の先頭－1）の位置
• キューのアドレス
• キューの長さ

これらはキューテーブルとしてひとまとめにして格納されています。キューテーブルは6バイトで、次のようになっています。

キューテーブル先頭	プット　オフセット	データの最後（キューの先頭を0とした位置）
＋1	ゲット　オフセット	空いている部分の最後（キューの先頭を0とした位置）
＋2	バック　キャラクタ	キューから文字を取り出すルーチンで、ここにデータ（≠0）があるとその文字をキューからのデータとする
＋3	キュー長	キューの長さ　2^n-1、$1≦n≦8$
＋4、＋5	キューアドレス	キューのアドレス

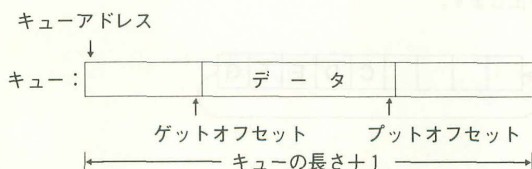
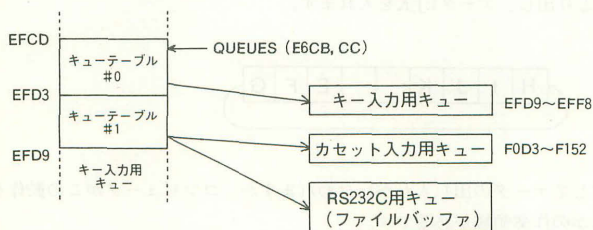



図5－4A

キューテーブルには＃0と＃1の2つがあり、＃0はキー入力用に、＃1はカセット入力とRS232C入力用に使われています。キューテーブル＃0のアドレスは、ポインタQUEUES（E6CB,CC)に入っています。普通はQUEUESの値はEFCDHです。

このキューを利用するとファンクションキーのような事ができます。詳しくは「第7章キー入力」を見て下さい。

第6章　カセット汎用入出力ポート

第6章　カセット・汎用入出力ポート

6-1. カセット・ファイル

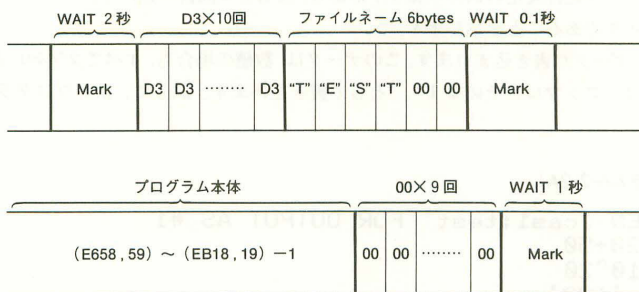
各ファイルにカセットを選択するには、デバイス名にcas1またはcas2を使います。添字を省略してcasとした時、および、ディスクシステムなしの場合のN_{88}-BASICでデバイス名を省略した時は、自動的にcas1が選択されます。つまり次の３つは全く同様の動作をします。

```
save "cas1:test"
save "cas:test"
save "test"     ........ディスクなしの場合のみ
```

なおcas1とcas2の違いはボーレイトの違いのみで、cas1は1200ボー、cas2はN-BASICモードの時と同じ600ボー、となっています。したがって、cas1を選択した時は、N-BASICモードの時より、データの転送速度が倍の速さになります。

6-2. データ・フォーマット

6-2-1 プログラム・ファイル

N_{88}-BASICプログラムをカセットにSAVEした時のフォーマットは、次の様になります。

（図6-2-1）

まずモータをONにし、テープ走行が安定するまで待ったあと、D3Hを10回書き込み、6バイトのファイルネームを書き込みます。ファイルネームが、6文字未満の時は、その後に00Hをつけ加え、6バイトにします。その後再びWAITがありますが、これは、LOADした時、この時間を利用して'Found'や'Skip'などの表示をしている為で、これがないと、オーバーランしてしまう恐れがあります。

そして、プログラム本体です。プログラム本体は、E658,59H番地に入っているプログラム開始番地から、EB18,19H番地に入っている番地の1つ前までを書き込みます。この本体の最後の3バイトは00Hになっていて、次に続く9回の00Hと合わせ、12回の00Hでエンドマークを作ります。

LOADの時、D3Hを10回読むと、プログラム・ファイルと見なし、ファイルネームの比較を行います。一致していれば、プログラムを読み込み、00Hが10回連続して読み出されると、プログラム・ファイルの終了と見なし、LOADを終了します。

ところで、このフォーマットは、N-BASICのフォーマットと、全く同一です。事実、メモリ容量を超えない範囲で、ボーレイトを600ボー（cas2）にした時、N_{88}モードでカセットにSAVEしたプログラムを、N-BAISCモードで読むことは可能ですし、その逆も可能です。しかし、いずれの場合も、実行することはできません。なぜなら、ステートメント機能が一部異なっていることや、中間言語のコードが違っているからです。

6-2-2 データ・ファイル

次にデータ・ファイルのフォーマットについて見てみましょう。

WAIT 2秒	9C × 6回						データ本体						WAIT 1秒
Mark	9C	9C	9C	9C	9C	9C	"7"	"0"	","	....	0D	0A	Mark

(図6-2-2)

最初にテープ走行安定の為のウェイトがあり、9CHを6回書き込んで、ヘッダを作り、データ・ファイルであることを示します。

続いて、データが書き込まれます。このデータは、数値の場合も、すべて文字列に直し、データの区切りのコンマは、そのまま"，"として書き込みます。試しに、次のプログラムを走らせると…

（プログラム6-2-2A）

```
10 OPEN "cas1:test" FOR OUTPUT AS #1
20 A=20+50
30 B=10^20
40 C$="123"
50 PRINT #1,A,B,C$
60 CLOSE
```

テープに書き込まれるデータのフォーマットは、次の様になります。

ヘッダ	" "	"7"	"0"	" "	","	" "	"1"	"E"	"+"	"2"	"0"	" "	","	"1"	"2"	"3"	0D	0A	Mark

(図6-2-3)

見てわかる様に、数値は画面に表示される時と全く同じ形の文字列に直され、数字の並びである文字列との区別はありません。したがって、カセットデータ・ファイルはすべて文字列だとも言えます。ここまでくるとわかると思いますが、読み込みの時、ファイルの内容が数値であろうと文字列であろうと、すべて文字変数で読めるのです。先のファイルを次のプログラムで読むと…

(プログラム6-2-2B)

```
10 OPEN "cas1:test" FOR INPUT AS #1
20 INPUT #1,A$,B$,C$
30 PRINT A$,B$,C$
run
70            1E+20          123
Ok
```

ちゃんと読めたでしょう。ですから、次の2つは全く同じ動作をします。

(プログラム6-2-2C)

```
 :
1000 INPUT #1,A,B
 :
```

(プログラム6-2-2D)

```
 :
1000 INPUT #1,A$,B$
1010 A=VAL(A$) : B=VAL(B$)
 :
```

ところでプログラム6-2-2Aの50行が次の様になっていたとすると…

(プログラム6-2-2E)

```
50 PRINT #1,A,B,C$;
```

画面と同じ様にCR（0DH），LF（0AH）が出力されず次の様なフォーマットになります。

(図6-2-4)

最後のデータをよく見て下さい。何の区切りもなくいきなり終わっています。このファイルをプログラム6-2-2Bで読んでみると…

```
run
?TP Error in 20
Ok
```

これは、読み込む時区切りがない為に、データが終わってもなお先を読もうとし、ファイルエンドのmark(＝データがない)に続くSpaceやノイズ等で、リードエラーを起こす為です。したがってプログラム6-2-2Eの様にセミコロンを最後につけると、最後のデータのみ読めなくなります。

この様に、カセットデータ・ファイルのフォーマットはN-BASICと同じですので、N88-BASICで、デバイス名にcas2（600ボー）を選択した時、双方のファイルは互いに読み込み可能です。詳しくは次節で述べましょう。

6-3. カセットデータ・ファイルのN-BASICとのコンパチビリティ

前節で述べた様に、N-BASICで作ったカセットデータファイルは、N88-BASICモードでも読むことができます。実際、次の２つのプログラムを各モードで走らせますと…

（プログラム6-3A）
N88-BASIC

```
10 OPEN "cas2:test" FOR OUTPUT AS #1
20 A$="ABCDE":B$="FGHIJ":C=5678
30 PRINT #1,A$,B$,C
```

（プログラム6-3B）
N-BASIC

```
10 A$="ABCDE":B$="FGHIJ":C=5678
20 PRINT #-1,A$,B$,C
```

でき上がったデータファイルは全く同じものとなり、それらを区別するのも難かしいのです。

ちょっと待って下さい。ところでN88-BASICモードの10行目のOPENで使ったファイル名は、どこへ行ったのでしょう。前節で見て来た様にファイル中には入っていません。

実は、このファイル名は、全く無視されたのです。ですから、使用可能な文字である限り、ファイル名は何にしようが同じことで、極端な場合、無くてもよいのです。当然、データファイルは、スキップされることなどなく、最初に出てきたファイルを読み込みます。プログラム6-3Aの10行を次の様に書き換えても、全く同じことです。

```
10 OPEN "cas2:ABCD" FOR OUTPUT AS #1
10 OPEN "cas2:" FOR OUTPUT AS #1
```

さて、ここまで述べますとカセットデータファイルに関しては、N-BASICと全く同じ様に見えますが、このファイルを読み込む時少々違った動作をすることがあります。いずれも、正しい使い方では起きない差ですが、デバックの途中などでは要注意です。

1） INPUTでファイルを読む時、入力する変数の数に比べファイル中のデータが少ない時、N-BASICでは“Out of Data”となったが、N88-BASICでは“TP Error”となる（プログラム6-3C，6-3D，ファイルはプログラム6-3Aで作成したもの）。

(プログラム6-3C)
N88-BASIC

```
10 OPEN "cas2:test" FOR INPUT AS #1
20 INPUT #1,A$,B$,C,D
30 PRINT A$,B$,C,D
40 CLOSE
run
?TP Error in 20
Ok
```

プログラム6-3D
N-BASIC

```
10 INPUT #-1,A$,B$,C,D
20 PRINT A$,B$,C,D
run
Out of DATA in 10
Ok
```

2) 逆に、入力する変数に比べファイル中のデータ数が多い時、N-BASICでは“Extra ignored”の表示が出たが、N88-BASICでは、何のメッセージも出ない（プログラム6-3E, 6-3F, ファイルはプログラム6-3Aによる)。

(プログラム6-3E)
N88-BASIC

```
10 OPEN "cas2:test" FOR INPUT AS #1
20 INPUT #1,A$,B$
30 PRINT A$,B$
40 CLOSE
run
ABCDE          FGHIJ
Ok
```

(プログラム6-3F)
N-BASIC

```
10 INPUT #-1,A$,B$
20 PRINT A$,B$
run
?Extra ignored
ABCDE          FGHIJ
Ok
```

3) 入力する変数が数値変数で、読んだファイルデータが数字以外の文字列の時、N-BASICでは、“Bad File Data”となったが、N88-BASICでは、エラーとならず、その変数に0が入っている。(プログラム6-3Gでファイル作成、プログラム6-3H、プログラム6-3Iで読み込み)。

(プログラム6-3G)

```
10 OPEN "cas2:test" FOR OUTPUT AS #1
20 PRINT #1,"123","ABCD","45ABC","&HABC"
30 CLOSE
```

(プログラム6-3H)
N88-BASIC

```
10 OPEN "cas2:test" FOR INPUT AS #1
20 INPUT #1,A,B,C,D
30 PRINT A,B,C,D
40 CLOSE
run
 123         0          45          2748
Ok
```

(プログラム6-3I)
N-BASIC

```
10 INPUT #-1,A,B,C,D
20 PRINT A,B,C,D
run
Bad File Data in 10
Ok
```

さて、これらの内、1)と2)の違いは、N-BASICとN_{88}-BASICとの、ファイルの取り込み方の違いによるものです。

N-BASICではファイル中の0DH（CR）を、ファイルエンドの印と見なし、これを見つけるまでは、読み込んだデータを全てバッファ内に取り込みます。その後、このバッファ内のデータは、キー入力の時と同様の扱いを受け、データの数、型などのチェックを受けるわけです。

ところが、N_{88}-BASICでは、ファイルエンドの印は在存せず、0DHは、単なるデータの区切りと見なします。読み込んだデータも区切りを見つけるたびに変数に代入され、また代入されるべき変数が残っていれば先を読み続け、すべての変数に代入し終えれば、ファイル中にデータが残っていても、そこで読み込みをやめてしまいます。

ですから、ファイル中のデータが足らなくてもお構いなしに先を読み、前節の例と同じ様にテープリードエラーを起こすのです。この時、録音が悪い為のエラーだと思って、何度テープレコーダのレベル調整や、ファイル作成をやり直しても無駄です。

またファイル中のデータが多過ぎる時は、PCは、モータも止めてしまい、まだデータが残っているなどとは夢にも思っていないかの様です。

そして、3)の違いですが、N-BASICでは、バッファ内からデータを変数に代入する時、型のチェックを行い、もし型が合わなければ、エラーとなります。ところが、N_{88}-BASICでは、チェックが行われないのです。代入すべき変数が文字変数であれば、どんなデータでも代入可能ですが、数値変数の時はどうするのでしょう？

そうです！ VAL関数を使うのです。次のプログラムと結果を見て下さい。

（プログラム6-3J）

```
10 A=VAL("123"):B=VAL("ABCD"):C=VAL("45ABC"):D=VAL("&HABC")
20 PRINT A,B,C,D
run
 123           0             45            2748
Ok
```

どうです？プログラム6-3Hと全く同じ結果でしょう。つまり、読み込んだデータに対し、VAL関数による評価を行い、その結果を変数に代入するのです。

以上が両モードの主な差違ですが、これ以外にもう1つ、デリミタ（区切り、終了符号）に関する問題があります。

データの区切りを示す符号として、両モード共に、コンマ“，”が使用されています。前節のフォーマットで示した様に、この“，”は、そのままの形で書き込まれ、読み込み時、“,”を見つけると無条件に、データを区切ってしまいます。従って、ファイルに文字列を書き込む時、データとして“,”を書き込んでも、読み込み時にそこで区切られ、順にデータがズレてしまいます。

次の様にファイルを作り、

(プログラム6-3K)

```
10 OPEN "cas2:" FOR OUTPUT AS #1
20 A$="ABCDE,FGHIJ" : B$="KLMN"
30 C$="OPQR" : D$="STUV"
40 PRINT #1,A$,B$,C$,D$
50 CLOSE
```

読み込ませると、こうなってしまいます。

(プログラム6-3L)

```
10 OPEN "cas2:" FOR INPUT AS #1
20 INPUT #1,A$,B$,C$,D$
30 PRINT A$,B$,C$,D$
40 CLOSE
run
ABCDE          FGHIJ          KLMN          OPQR
Ok
```

ファイルデータとして“,”を使いたければ、LINE INPUTを使えばよいでしょう。

(プログラム6-3M)

```
10 OPEN "cas2:" FOR INPUT AS #1
20 LINE INPUT #1,A$
30 PRINT A$
40 CLOSE
run
ABCDE,FGHIJ,KLMN,OPQR,STUV
Ok
```

そして、もう1つのデリミタ(終了符号)ですが、前に述べた様に、N-BASICでは読み込み時に0DH (CR) を見つけると、ファイルの終了と見なし読み込みをやめます。ところが、N_{88}-BASICでは、終了符号に相当するものがなく、0DHは単なる区切りと同等に扱われます。ですから、文字列中にCHR$(13)を含んだ場合は、N-BASICではデータの不足になる恐れがあり、N_{88}-BASICでは“,”と同様、データがズレる恐れがあります。

（プログラム6-3N）

```
10 OPEN "cas2:" FOR OUTPUT AS #1
20 A$="ABCDE" : B$="FGHIJ"+CHR$(13)+"KLMN"
30 C$="OPQR" : D$="STUV"
40 PRINT #1,A$,B$,C$,D$
50 CLOSE
```

（プログラム6-3O）
N88-BASIC

```
10 OPEN "cas2:" FOR INPUT AS #1
20 INPUT #1,A$,B$,C$,D$
30 PRINT A$,B$,C$,D$
40 CLOSE
run
ABCDE         FGHIJ         KLMN          OPQR
Ok
```

（プログラム6-3P）
N-BASIC

```
10 INPUT #-1,A$,B$,C$,D$
20 PRINT A$,B$,C$,D$
run
Out of DATA in 10
Ok
```

6-4. 汎用入出力ポート

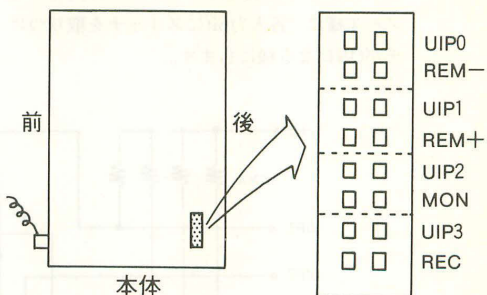
PC-8801には、汎用入出力ポートがあり、4bitの入力、1bitの出力がユーザに開放されています。

この入出力ポートは、カセットインターフェイス用コネクタに引き出されており、入力ポートは、内部のジャンパスイッチによって、カセット信号線と切り換える様になっています。当然ながら、入力ポートに切り換えた時、その端子に対応するカセット端子が使えず、それを使う命令も使えなくなります。残念ながら、この切り換えはソフトによる制御ができません。

出力ポートの方は、常にコネクタに引き出されており、カセットとの同時使用が可能です。

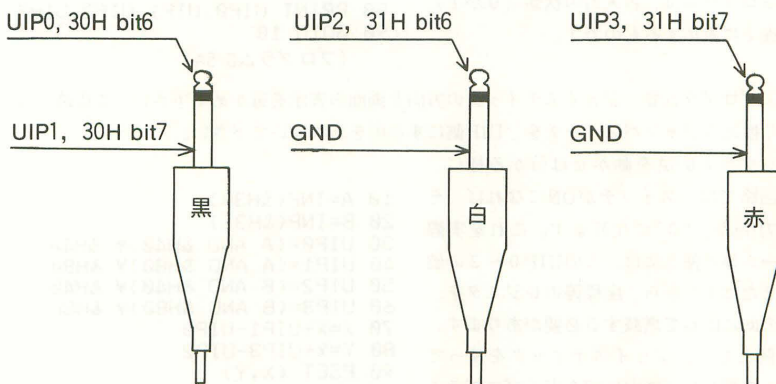
コネクタのピン配置と入力切り換えジャンパスイッチの対応、および、入力に切り換えた時の入力ビットとカセットプラグの関係を示します。なお、コネクタの3番ピン$\overline{\text{INT5}}$は使用できません。

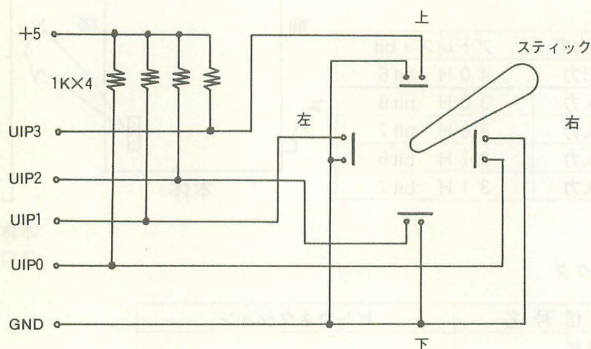
アドレス対応表

信号名	入出力	アドレス・bit
UOP0	出力	40H　bit6
UIP0	入力	30H　bit6
UIP1	入力	30H　bit7
UIP2	入力	31H　bit6
UIP3	入力	31H　bit7

本体の向って右後、
スロットバスの下

カセット用コネクタ

端子番号	信号名	ピンコネクション
1	＋5V	
2	GND	
3	$\overline{\text{INT5}}$	
4	REC/UIP3	
5	MON/UIP2	
6	REM＋/UIP1	
7	REM－/UIP0	
8	UOP0	

カセットプラグ対応図

6-5. ジョイスティックの接続

入力ポートは4bitあるので、4方向のジョイスティックを接続するのは簡単で、図6-5に示す様に、各入力bitにスイッチを取りつけ、スティックを倒した時に、その方向のスイッチがONになる様にします。

(図6-5)

ジョイスティックの接続はこれだけです!? ここでやめてしまうのは、いささか早計で、接続したのはいいけれど、まだ読み込みの方法が分かりません。そこで、次は、このジョイスティックの方向を読みとるソフトが必要になります。

入力ポートは先に示した様に、I/Oポートの30H、31Hのbit6とbit7にあり、その状態をBASICで読みとるには、INP関数を使い、その後各bitを抽出しなければなりません。右のプログラムは、各入力の状態（0か1）を画面上に表示するものです。

```
10 A=INP(&H30)
20 B=INP(&H31)
30 UIP0=(A AND &H40)¥ &H40
40 UIP1=(A AND &H80)¥ &H80
50 UIP2=(B AND &H40)¥ &H40
60 UIP3=(B AND &H80)¥ &H80
70 LOCATE 0,13
80 PRINT UIP0,UIP1,UIP2,UIP3
90 GOTO 10
```

(プログラム6-5A)

このプログラムで、ジョイステイックの方向と画面の表示を確かめて下さい。この時、入力切り換えのジャンパスイッチを、UIP側にするのを忘れないで下さい。

プログラム6-5Aを動かせば分かる様に、先の回路では、スイッチがONになれば、その入力bitが、“0”になります。これを実際のゲーム等に使うには、このUIP 0～3の値（0または1）から、座標等のレジスタを、その方向に応じて増減する必要があります。その例として、ジョイスティックを使ってドットを動かし、画面に線を引くプログラムを示します。

```
10 A=INP(&H30)
20 B=INP(&H31)
30 UIP0=(A AND &H40)¥ &H40
40 UIP1=(A AND &H80)¥ &H80
50 UIP2=(B AND &H40)¥ &H40
60 UIP3=(B AND &H80)¥ &H80
70 X=X+UIP1-UIP0
80 Y=Y+UIP3-UIP2
90 PSET (X,Y)
100 IF INKEY$="e" THEN CLS 3
110 GOTO 10
```

ところで、ジョイスティックを使う為に、切り換えのジャンパスイッチをUIP側にすると、カセットインターフェイスが使えず、プログラムのカセットへのSAVE、LOADなどができません。これでは、ディスクを持っていない人は、ゲームプログラムを読み込むことすらできません。これでは、いくらジョイスティックを接続しても意味がありません。そこで、不十分ではありますが、その対策の１例をあげておきます。

プログラムのSAVE、LOADを行うには、REC端子（赤いプラグ）とMON端子（白いプラグ）があればよく、REM端子（黒いプラグ）はなくても可能です。ただ、REM端子を使わない時、テープ走行は自動的に止まらず、そのつど、テープレコーダを自分の手で止めなければなりませんが…。このREM端子を使わないことにして、この部分だけ、ジャンパスイッチをUIP側にすると、REM－とREM＋だった黒プラグは、UIP０とUIP１になります。

こうすると、プログラムのSAVE、LOADもできますし、入力も2bitのみ可能です。この時、MOTOR命令は意味を持たなくなり、４方向のジョイスティックが使えず、２方向のみの入力となりますが、特別な回路を組まずに済ますには、しかたないでしょう。

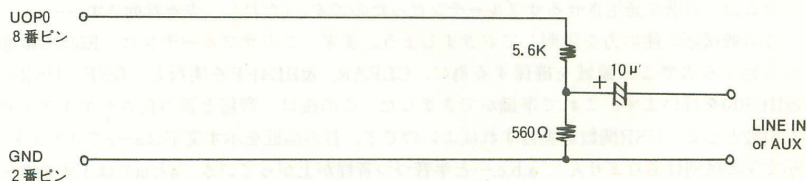
6-6. 出力ポートによる効果音

出力ポートは、I/Oポートの４０H番地bit6で、この1bitのみしかありません。しかし、入力ポートの時の様に切り換え式でなく、常にコネクタピンに出力されているので、出力ポートを使用しても、カセットインターフェイスの使用制限は起こりません。

使用の際には、この出力ピンはカセットケーブルに接続されていないので、ケーブルに一本電線を増しピンに接続するか、コネクタを別に用意し専用ケーブルを作るかしなければなりません。

このポートの最も簡単な利用法として、音を出すことに使って見ましょう。

まず、次の回路図（図６-６-１）の様な簡単な減衰器を作り、一方をUOP0に、もう一方をアンプやラジオカセット等のLINE入力、AUX端子等に接続します。

（図6-1-1）

接続が終わったら、次のプログラムを実行させ、音をモニタしてみて下さい。

```
10 FOR I=0 TO 499
20 OUT &H40,PEEK(&HE6C1) OR &H40
30 OUT &H40,PEEK(&HE6C1) AND &HDF
40 NEXT I
```

音が出ましたか？ もし出ない時は、プログラムや、配線をよく確めて下さい。

いま出した音は、単に“ブー”といった音で全然面白味がありません。BASICで作った場合、処理速度の関係でこんな音しか出せないのです。そこで、この音を出すプログラムを機械語で書いてみましょう。

モニタモードで、次のプログラムを入れて下さい。

```
E500 CD FC 56 CD BD 40 57 3E 03 14 3C 5F 15 C8 7E 23
E510 D6 31 FE 09 38 F4 D6 10 E6 DF FE 1A D2 93 03 E5
E520 D5 21 42 E5 87 4F 06 00 09 4E 23 56 F3 3A C1 E6
E530 EE 40 CD CD 3E 41 10 FE 15 20 F1 1D 20 ED D1 E1
E540 18 CA F8 2E EA 32 DD 34 D0 38 C4 3A B8 3E AE 42
E550 A4 46 9A 4A 91 4E 88 54 80 58 79 5E 72 64 6B 6A
E560 65 70 5F 76 59 7E 53 86 4E 8E 4A 94 45 9E 41 A8
E570 3D B2 39 BC 36 C6 00 00 00 00 00 00 00 00 00 00
```

入れ終わったら、BASICモードに戻して、次のプログラムを実行して見て下さい。

```
10 CLEAR 0,&HE4FF : DEF USR=&HE500
20 OUT 104,0
30 A$="ceghjlm"
40 B$="ijhgeca"
50 A$=USR("1"+A$+B$+A$+B$)
60 WIDTH 80
```

さっきの音とは違った、変わった音が、出たでしょう。実は、さっき入れた機械語のプログラムは、音階を発生させるサブルーチンだったのです。(ただし、少々音痴ですが…)

この機械語の使い方を説明しておきましょう。まず、このサブルーチンは、E500H番地から始まるのでこの領域を確保する為に、CLEAR，&HE4FFを実行し、DEF USR＝&HE500を行います。これで準備ができました。この後は、音程と音の長さを示す文字列を引数として、USR関数を実行すればよいのです。音の高低を示す文字はa～zで（大文字、小文字の区別はありません）、a,b,c,…と半音づつ音程が上がっていき、aとmでは１オクターブの関係になります。長さは１～９の数字で、数に比例した長さとなります。その他の文字を含んだ場合、“SN Error”となります。

次のプログラムを実行させて下さい。ある曲が流れてきます。

```
10 CLEAR 0,&HE4FF : DEF USR=&HE500
20 A$="8fhjjfhjj6m8jhf"
30 B$="hjhh"
40 C$="hjff"
50 OUT 104,0
60 A$=USR(A$+B$+A$+C$)
70 WIDTH 80
```

ところで50行のOUT命令は何でしょう。これは、画面表示の為のDMAをストップさせているのです。なぜストップさせるか、と言いますと、これがないと音がにごってしまうからです。試しに50行をとって実行させて見て下さい。なお、DMAをストップさせると、画面に何も表示されなくなり、もしエラーが出ても何のことかさっぱり分からなくなりますので、この行は最初REM文にしておいて、実行させてエラーがないことを確認した後、REMを抜くとよいでしょう。DMAを再開させ画面を表示するには、WIDTH 80またはWIDTH 40を実行させればOKです。

第7章　キー入力

第7章　キー入力

7-1. キー入力バッファ

キー入力バッファは第5章で述べたようにキュー形式をとっています。キュー長は31文字で、このため31文字までの先行入力ができます。

キューアドレス：ＥＦＤ９〜ＥＦＦ８Ｈ
キューテーブル
　プットオフセット：ＥＦＣＤＨ
　ゲットオフセット：ＥＦＣＥＨ
　バックキャラクタ：ＥＦＣＦＨ
　キュー長　　　　：ＥＦＤ０Ｈ
　キューアドレス　：ＥＦＤ１Ｈ，ＥＦＤ２Ｈ

これを利用して、プログラム中で、ある文字列をあたかもキーボードから打ったような動作をさせることができます。

1)　文字列の長さが31文字以内のとき。
- キューの先頭（EFD9H番地）から文字列を書き込みます。
- プットオフセット（EFCDH番地）に文字列の長さ－1を入れます。
- ゲットオフセット（EFCEH番地）に1FHつまり31を入れます。

こうすると、キー入力待ちになった時に、その文字列をキーボードから打ち込んだのと似た動作をします。

実際にやってみましょう。

```
list
10 A$="Tech-Know 8800    SYSTEM SOFT "
20 FOR I=1 TO LEN(A$)
30  POKE &HEFD8+I,ASC(MID$(A$,I,1))
40 NEXT
50 POKE &HEFCD,LEN(A$)-1
60 POKE &HEFCE,31
Ok
run
Ok
Tech-Know 8800    SYSTEM SOFT
```

E6CDH に 255 を入れると、キー入力を受けつけなくなります。こうしておかないと、このプログラムを実行中にキーを押すとキューテーブルがくるってしまうからです。

ところで、このようにソフト的にファンクションキーの動作をさせるのは、もっと簡単にできます。第12章ランダムテクニックの「ソフトファンクションキー」を見て下さい。

2) 文字列の長さが32文字以上のとき。

- メモリ上の適当な場所に文字列を書き込みます。
- プットオフセットに文字列の長さ－1を入れます。
- キュー長（EFD0H番地）に、2^n-1を満たし文字列の長さよりも大きな値を入れます。例えば40字の文字列の場合は$2^6-1=63$です。
- キュー長と同じものをゲットオフセットにも入れます。
- キューアドレス（EFD1,2H番地）に文字列の先頭アドレスを入れます。

つまりキューの位置を移動するわけですが、動かしたままではいけませんので、あとで元に戻してやる必要があります。これは、35D9H番地をCALLすることによって簡単に行うことができます。

キューの移動ということで面白いことをやってみましょう。

```
POKE &HEFD1,&HC8 : POKE &HEFD2,&HF3 : CONSOLE 1,25
```

を実行します。キューを画面上に移したわけです。何かキーから入力すると、画面の左上に同じものが現われます。コントロールキーやカーソル移動キーを押すと対応するキャラクタが現れます。キューを元に戻すにはSTOPキーを押します。

7-2. ファンクションキー

ファンクションキーの内容はE6F2H～E791H番地に格納されています。

F・1	E6F2H～E701H	F・6	E742H～E751H
F・2	E702H～E711H	F・7	E752H～E761H
F・3	E712H～E721H	F・8	E762H～E771H
F・4	E722H～E731H	F・9	E772H～E781H
F・5	E732H～E741H	F・10	E782H～E791H

各キーにつき16バイトが割り当てられていますが、ターミネータとして00が必要ですので、定義できるのは15文字までとなっています。

ファンクションキーを実現するアルゴリズムは簡単で、ファンクションキーが押されると、定義されている文字列をキューにほうり込むだけです。もしキューが一杯になったら残りの文字は無視されます。

さきほどターミネータとして00が必要だと述べましたが、これをなくしてしまうとどうなるでしょう。

```
key 1,"ABCDEFGHIJKLMNO"
Ok
key 2,"0123456789"
Ok
poke &HE701,ASC("@")   ←ターミネータの所に@を書き込む
Ok
```

ここで f・1 を押します

```
ABCDEFGHIJKLMNO@0123456789
```

f・1とf・2がつながってしまいました。こうすることによって16文字以上の定義もできます。しかしキューの長さが31文字ですので32文字以上の文字列を定義しても無意味です。

ファンクションキーの初期データはN_{88}-BASIC ROM内の1B0H～24FHに入っています。電源ON(リセット)の時にこのデータがE6F2H～E791Hに転送されて来るわけです。従がってファンクションキーを初めの状態に戻すには、もう一度データを転送してやればよいわけです。これをBASICで行なうと次のようになります。

```
100 '
110 ' --- function key init sub
120 '
130 *F.INIT
140 FOR I=0 TO 159
150   POKE &HE6F2+I,PEEK(&H1B0+I)
160 NEXT
170 RETURN
```

これはサブルーチンになっていて、必要な所にGOSUB *F・INITを入れておくとファンクションキーが初期化されます。しかし表示は変わりませんのでシフトキーを押すか、CONSOLE ,,1を実行して下さい。

次にこれを機械語で行なった例を示します。

```
100 '
110 ' Function key INIT Command
120 '
130 FOR I=&HF260 TO &HF270
140   READ D$ : POKE I,VAL("&H"+D$)
150 NEXT
160 POKE &HEEB7,&H60 : POKE &HEEB8,&HF2
170 DATA E5,21,B0,01,11,F2,E6,01,A0,00,ED,B0,CD,79,
         3F,E1,C9
```

このルーチンを実行すると、新しいコマンド'CMD'ができ、他のステートメントと同じように使えます。プログラム上の必要な箇所にCMDと入れれば良いわけです。

7-3. キー入力ステートメント活用テクニック

7-3-1 INPUT文と疑問符

N-BASICでのINPUT文では、入力待ちになる時、プロンプト文に続けて'? 'が出力されますが、N_{88}-BASICでは、出力させなくすることもできるようになっています。input コマンドを入力する際、プロンプト文の後に“,”を入れて下さい。

これは、プログラムを作る上でも便利なもので、たとえば、16進数を入力させる場合、次のようなプログラムにすることもできます。

```
list
10 INPUT "Start Address .. &H",SA$
20 S.ADRS=VAL("&H"+SA$)
Ok
run
Start Address .. &H■
                   └─カーソル

run
Start Address .. &H44a5
Ok
```

7-3-2 INPUT WAIT文と待ち時間

INPUT WAIT文は待ち時間だけキーボードからの入力を待つINPUT文です。普通のINPUT文は、このWAITが無限大のものと思えば良いでしょう。

この待ち時間は、(INPUT WAITの後で指定した値）×0.1秒ということになっていますが、どのくらいの大きさまで使えるのでしょうか？

実際にやってみると、32768以上で'OV'エラーになってしまいますね。

つまり、待ち時間というのは、整数型の数値または変数でなければならないわけで、もしそれ以外（10.3など）であれば、整数型に変換され、それが待ち時間となります。

となると、最大待ち時間は3276.7秒ということになりますが、本当にそうでしょうか。実は、この値には、負の数も使えるのです。(整数型というのは－32768～32767というのを思い出して下さい）試しに、INPUT WAIT －1,Aとすると6553.5秒間待つことになります。

なお、待ち時間を０（0.4などでも同様）にすると、FCエラーとなりますので注意して下さい。

また、このINPUT WAIT文は、キー入力がないということを前提とすると、一種のタイマーとして使うこともできます。

```
100 INPUT WAIT 10,"",
110 PRINT I,
120 I=I+1
130 GOTO 100
```

7-3-3 LINE INPUT文と数値の代入

INPUT文では、入力するデータの個数や型が違っていると"? Redo from Start"などというメッセージが出力され、時には画面がこわされたりすることがあります。これをなんとかしようということで登場して来るのが、このLINE INPUT文です。

次の例は、名前と年令を","で区切って入力させるサブルーチンですが、入力ミスがある場合は、画面をこわすことなく再入力させることができます。

```
100 *D.INPUT
110  LOCATE 0,10 : LINE INPUT "ﾅﾏｴ , ﾈﾝﾚｲ ? ",LN$
120  SP=INSTR(LN$,",") : IF SP<2 THEN *ER
130  NM$=LEFT$(LN$,SP-1)
140  AGE=VAL(MID$(LN$,SP+1))
150  IF NM$<>"" AND AGE>0 THEN *OK
160 '
170 *ER
180  LOCATE 0,10 : PRINT "... ﾏﾁｶﾞｲ ...";STRING$(9,CHR$(9));
190  BEEP
200 GOTO *D.INPUT
210 '
220 *OK
230  PRINT "ﾅﾏｴ  ... ";NM$
240  PRINT "ﾈﾝﾚｲ ... ";AGE
250 END
```

INPUT　NM$,AGEと書くよりもかなり複雑なものになっていますが、不特定多数の人に使われるプログラム（ビジネス・アプリケーション）などでは、これくらい注意を払ったプログラムにしたいものです。

7-3-4 INP関数とWAIT文

INP関数およびWAIT文を使うと、直接入力ポートを通して、キーボードの状態を知ることができます。

INP関数は、機械語のIN命令と同じものです。

A=INP(PORT)　　=　　IN A,(PORT)

この関数を使うと、INKEY$などに比べて高速のキーセンスが可能で、ゲームなどにはもってこいですね。

```
100 ' key sence < TEN key >
110 *LOOP
120  PT0=(NOT INP(0) AND &HFF)
130  PT1=(NOT INP(1) AND &HFF)
140   IF PT0 AND    4 THEN PRINT "DOWN"
150   IF PT0 AND &H10 THEN PRINT "LEFT"
160   IF PT0 AND &H40 THEN PRINT "RIGHT"
170   IF PT1 AND    1 THEN PRINT "UP"
180 GOTO *LOOP
```

WAIT文は、INP関数の応用と考えるとよいでしょう。INP関数を使ってWAIT文を書くと次のようになります。

```
100 ' WAIT statement  :  WAIT port,val1,val2
110 *LOOP
120  A=INP(PORT)
130  IF (A XOR VAL2) AND VAL1)=0 THEN *LOOP
```

なお、INP,WAITを使った場合でも、押されたキーのデータは、キーバッファに入ってしまいます。これをクリアする方法は、7-3-5を見て下さい。

7-3-5 キーバッファのクリア

N_{88}-BASICはキーの先行入力が可能です。これはたいへん重宝な機能なのですが、反面困ることもあります。知らず知らずのうち☑キーを押してしまってあとであわてた経験はどなたでもおありでしょう。これがダイレクトモードであればSTOPキーを押せば良いのですが、プログラム実行中だとこういうわけにも行きません。あらかじめ適所適所でキーバッファをクリアしてやらなければなりません。

キーバッファのクリアをBASICで行なうとこうなります。

```
*KEY.CLEAR : IF  INKEY$ <> "" THEN *KEY.CLEAR
```

または、

```
WHILE INKEY$ <> "" : WEND
```

この機能をROM内のルーチンをCALLすることにより簡単に、かつ素早く行なうこともできます。アドレスは35D9Hですから

```
DEF USR=&H35D9 : A=USR (0)
```

または

```
KEY.CLEAR=&H35D9 : CALL KEY.CLEAR
```

とします。

7-3-6 INKEY$でカーソル表示

INKEY$関数でキーセンスを行う場合、カーソルを表示させる方法を紹介します。
カーソルをON/OFFする方法は次の通りです。

• カーソルON

```
DEF USR=&H4290 : A=USR(0)
```

• カーソルOFF

```
DEF USR=&H428B : A=USR(0)
```

それでは、これを使った例を見てみましょう。

```
100 ' INKEY$ with cursor
110 *LOOP
120 GOSUB *C.ON
130  A$=INKEY$ : IF A$="" THEN 130
140  PRINT A$,
150 GOTO *LOOP
160 END
1000 ' Cursor ON
1010 *C.ON
1020  DEF USR=&H4290 : A=USR(0)
1030 RETURN
2000 ' Cursor OFF
2010 *C.OFF
2020  DEF USR=&H428B : A=USR(0)
2030 RETURN
```

なおこの方法は、INP関数,WAIT文などでも用いることができます。

★文字列の入力方法の比較表

		INPUT	INPUT WAIT	LINE INPUT	LIINE INPUT WAIT	INPUT$(X)	INKEY$
		文	文	文	文	関数	関数
入力表示	プロンプト文	可能	可能	可能	可能	不可能	不可能
	プロンプトマーク?	可能	可能	無	無	無	無
	カーソル表示	有	有	有	有	有	可能*1
	エコーバック	有	有	有	有	無	無
	入力待ち	待つ	指定した時間だけ待つ	待つ	指定した時間だけ待つ	待つ	待たない
データ入力	,の入力	ダブルクォートで囲めば可	ダブルクォートで囲めば可	可能	可能	可能	可能
	"の入力	文字列の最初でなければ可	文字列の最初でなければ可	可能	可能	可能	可能
	コントロールコード カーソルキーの入力	不可	不可	不可	不可	可能	可能
	複数の変数への入力	可能	可能	不可	不可	不可	不可
	入力文字数	254文字以内	254文字以内	254文字以内	254文字以内	指定した文字数(255文字以内)	1文字
	入力終了	⏎キー	⏎キー	⏎キー	⏎キー	自動	——
	入力文字なしで⏎キー	ヌルストリング	ヌルストリング	ヌルストリング	ヌルストリング	CHR$(13)	CHR$(13)
STOPキー	BREAK表示	あり	あり	あり	あり	なし	あり*2
	CONTによる再開	可能	可能	可能	可能	不可	可能*2

*1：通常はカーソルは表示されませんが、表示させることも可能です。本文を見て下さい。

*2：INKEY$がブレークされたりCONTされたりするわけではありません。
INKEY$自体はSTOPキーをCHR$(3)として受けつけることも可能です。

★キーセンス比較表

	INPUT$(1)	INKEY$	INP(X)	WAIT
入力待ち	待つ	待たない	待たない	待つ
STOPキー	中断(CONT不可)	中断*	中断*	中断しない
複数キーの同時入力	不可	不可	可能	可能な場合もある
入力文字の判断	簡単	簡単	複雑	複雑
カーソル表示	表示する	表示させることも可	表示させることも可	表示させることも可
キー割り込み	きかない	きく*	きく*	きかない

*入力待ちがないためINKEY$, INP(X)自体はSTOPキーやキー割り込みとは関係ありません。

第8章　プリンタ（PC-8023＆PC-8821/22）

第8章　プリンタ(PC-8023&PC-8821/22)

8-1. 画面コピー

N88-BASICでは、COPY文とCOPYキーにより、画面上に表示されている文字や図形をプリンタに出力する機能を持っています。

テキストとグラフィックという2つの画面及び640×200ドット、640×400ドットという2つのグラフィックモードに対応して、コピーの形式もいくつか用意されています。

この節では、コピー機能の様々な使い方とその応用について説明していきましょう。

8-1-1 COPY文とCOPYキー

COPY文の機能については、マニュアルにあるように5種類のものがあります。

また、COPYキーの操作については、CTRLキー及びGRPHキーとの組み合わせにより、3通りのやり方があります。

このCOPY文とCOPYキーの対応を含めて実際の画面コピーの例を見てみましょう。

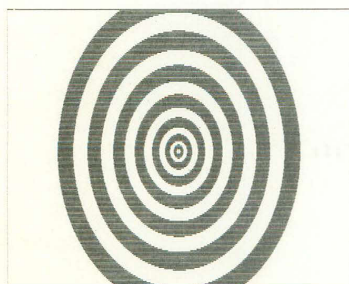
画面コピーテストプログラム（640×200モード　例1）

```
1000 '
1010 '  SCREEN COPY TEST PROGRAM ( EX.1 )
1020 '
1030 CONSOLE 0,25,0 : WIDTH 40,25 : SCREEN 0,0
1040 '
1050 SCREEN ,3 : CLS 3 : SCREEN ,0
1060 LINE(0,0)-(639,199),7,B
1070 FOR I=1 TO 15
1080   CIRCLE(320,100),I*I
1090 NEXT
1100 FOR I=2 TO 15 STEP 2
1110   PAINT(319-I*I,99),5,7
1120 NEXT
1130 '
1140 FOR I=0 TO 24
1150   LOCATE 17,I : PRINT USING "LINE ##";I;
1160 NEXT
1170 LOCATE  0, 0 : PRINT "■( 0, 0)";
1180 LOCATE  0,24 : PRINT "■( 0,24)";
1190 LOCATE 32, 0 : PRINT "(39, 0)■";
1200 LOCATE 32,24 : PRINT "(39,24)";
1210 POKE &HFF56,ASC("■")
1220 LOCATE 0, 3
1230 END
```

画面コピーテストプログラム　(640×400モード　例２)

```
1000 '
1010 '  SCREEN COPY TEST PROGRAM ( EX.2 )
1020 '
1030 CONSOLE 0,25,0 : WIDTH 40,25 : SCREEN 2,0
1040 '
1050 SCREEN ,3 : CLS 3 : SCREEN ,0
1060 LINE(0,0)-(639,399),7,B
1070 FOR I=1 TO 15
1080   CIRCLE(320,200),I*I
1090 NEXT
1100 FOR I=2 TO 15 STEP 2
1110   PAINT(319-I*I,199),5,7
1120 NEXT
1130 '
1140 FOR I=0 TO 24
1150   LOCATE 17,I : PRINT USING "LINE ##";I;
1160 NEXT
1170 LOCATE  0, 0 : PRINT "■( 0, 0)";
1180 LOCATE  0,24 : PRINT "■( 0,24)";
1190 LOCATE 32, 0 : PRINT "(39, 0)■";
1200 LOCATE 32,24 : PRINT "(39,24)";
1210 POKE &HFF56,ASC("■")
1220 LOCATE 0, 3
1230 END
```

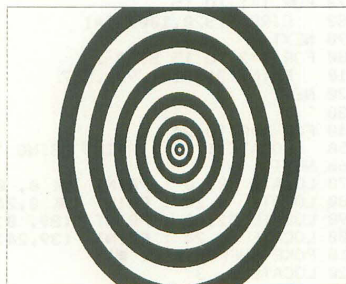
例１

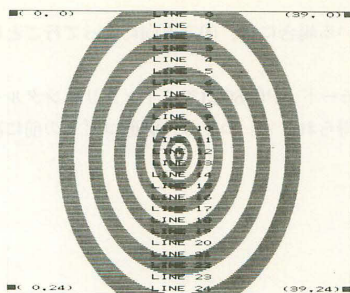
```
■( 0, 0)            LINE  0            (39, 0)■
                    LINE  1
                    LINE  2
                    LINE  3
                    LINE  4
                    LINE  5
                    LINE  6
                    LINE  7
                    LINE  8
                    LINE  9
                    LINE 10
                    LINE 11
                    LINE 12
                    LINE 13
                    LINE 14
                    LINE 15
                    LINE 16
                    LINE 17
                    LINE 18
                    LINE 19
                    LINE 20
                    LINE 21
                    LINE 22
                    LINE 23
■( 0,24)            LINE 24            (39,24)■
```

＝CTRL ＋ COPY

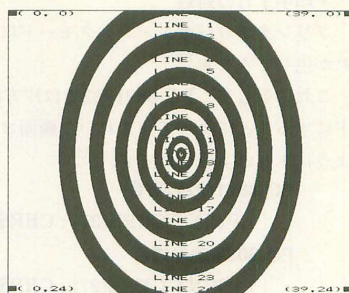
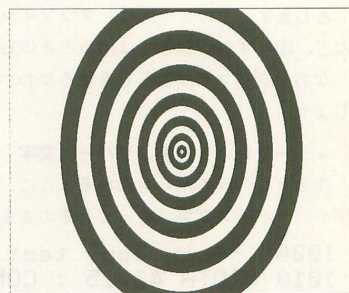
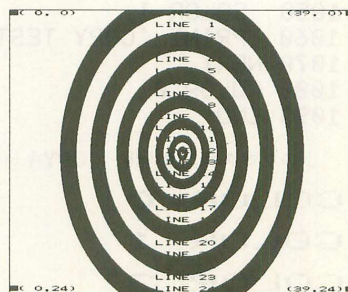
＝GRPH ＋ COPY

例２

```
■( 0, 0)            LINE  0            (39, 0)■
                    LINE  1
                    LINE  2
                    LINE  3
                    LINE  4
                    LINE  5
                    LINE  6
                    LINE  7
                    LINE  8
                    LINE  9
                    LINE 10
                    LINE 11
                    LINE 12
                    LINE 13
                    LINE 14
                    LINE 15
                    LINE 16
                    LINE 17
                    LINE 18
                    LINE 19
                    LINE 20
                    LINE 21
                    LINE 22
                    LINE 23
■( 0,24)            LINE 24            (39,24)■
```

＝CTRL ＋ COPY

＝GRPH ＋ COPY

例1

= COPY

例2

= COPY

= CRPH + COPY

= COPY

8-1-2 画面コピーを用いるときのテクニック

• 印字ずれの対処

プリンタがロジカル・シークモードになっている場合には、印字方向によって行ごとにずれが生じてきます。

これに対して、PC-8821/22では片方向印字モード、PC-8023ではインクリメンタルモードにすることによって、きれいな画面コピーが得られます。つまり、画面コピーの前に次のようにすればよいわけです。

PC-8821/22の場合

LPRINT CHR$(27) ; CHR$(62)

PC-8023(C)の場合

LPRINT CHR$(27) ; CHR$(91)

これらのモードの解除は、どちらの場合も

LPRINT CHR$(27) ; CHR$(93)

とします。上のモードは、グラフィック・キャラクタで表を作成したものを印字するときなど、印字ずれが起きては困るときにも使えますので、覚えておくと便利です。

なお、プリンタがどちらを使うかわからない場合は、上の2つとも実行しておけば良いでしょう。

• シークレット文字、リバース文字

次のプログラムからもわかるように、テキスト画面のコピーでは、画面上では見えない文字や、リバース文字であってもそのまま普通の文字で印字されます。

```
1000 '----- copy test
1010 WIDTH 40,25 : CONSOLE 0,25,1,0
1020 FOR I=0 TO 7
1030  COLOR 0
1040  PRINT "COLOR";I,
1050  COLOR I
1060  PRINT "COPY TEST"
1070 NEXT
1080 COLOR 0
1090 END
```

上のプログラム実行後、COPYキーで画面コピーしたもの

```
COLOR 0          COPY TEST
COLOR 1          COPY TEST
COLOR 2          COPY TEST
COLOR 3          COPY TEST
COLOR 4          COPY TEST
COLOR 5          COPY TEST
COLOR 6          COPY TEST
COLOR 7          COPY TEST
Ok
```

• 真円のコピー

真円をコピーするとたて長になってしまいますが、これもどうしようもないですね。自分で新たに画面コピープログラムを作って対処するしかありません。参考までに8-1-3で、グラフィック画面コピープログラムをとりあげていますので、これをもとに自分なりの画面コピープログラムを作ってみて下さい。なお、このプログラムでは1：1.03の比で真円がそのままコピーされます。

8-1-3 カラーグラフィック画面コピープログラム

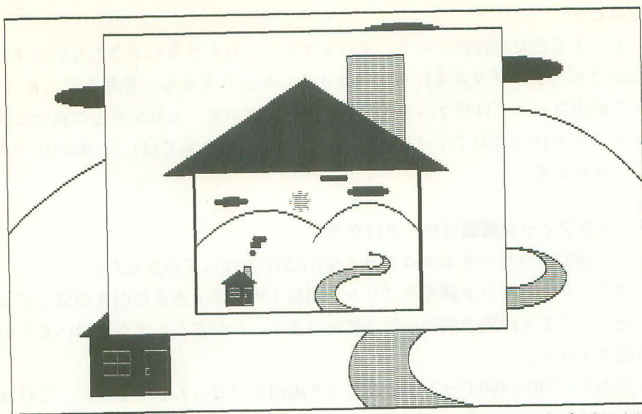
グラフィック画面をコピーするプログラムをBASICで作ってみました。

これは、カラーグラフィック画面をプリンタ用紙1枚分の大きさに引きのばしてコピーするもので、色に応じて4段階の濃淡が付けてあります。ただどうしようもないくらい遅いので実用には向きません。

テキスト画面や、640×400ドットグラフィック画面のコピーもできません。これは読者の方への課題としておきましょう。

```
5000 '----- CRT COPY PROGRAM  ( 640 × 200 color mode)
5010 *COPY.GRPH
5020 WIDTH LPRINT 255       (…8-4参照)
5030 '
5040 ESC$=CHR$(27)
5050 LPRINT ESC$"M";
5060 LPRINT ESC$">";
5070 LPRINT ESC$"T16";
5080 '
5090 FOR X=0 TO 79
5100  LPRINT ESC$"S0800";
5110  FOR Y=199 TO 0 STEP -1
5120   FOR C=1 TO 4 : D(C)=0 : NEXT C
5130   AP=1
5140   FOR B=0 TO 7
5150    PO=INT((POINT(X*8+B,Y)+1)/2)
5160    IF PO=0 THEN 5200
5170    FOR C=1 TO PO
5180     D(C)=D(C)+AP
5190    NEXT C
5200    AP=AP+AP
5210   NEXT B
5220   FOR C=1 TO 4
5230    LPRINT CHR$(D(C));
5240   NEXT C
5250  NEXT Y
5260  LPRINT
5270 NEXT X
5280 RETURN
```

このプログラムはサブルーチンとして使います。

コピーしたいところで、GOSUB ＊COPY. GRPHとします。

（NEC デモプログラムより）

8-2. PRINT文の出力をプリンタに

PRINT文によって画面に出力するプログラムで、その出力先をプリンタに変更したいとき、あなたはどうしていますか。

プログラムリストをながめつつ、いちいちPRINT文をLPRINT文に直していくというようなことをやってはいませんか。最終的にLPRINT文になってしまって良い場合はともかく、再びLPRINT文をPRINT文にするなど、どうも能率的ではありません。

また、LPRINT文を多用したプログラムのデバッグのために、いちいちプリンタ用紙へ出力していては紙の無駄でもあります。

それから、画面とプリンタへ同じものを出力するのに2通りの同じような文を書いたりしていませんか（次のプログラム例）。

```
1000 ' ----- MEMORY DUMP -----
1010 WIDTH 80,25
1020 DEFINT A-Z
1030 '
1040 INPUT "START ADRS ? &H",SA$
1050 INPUT "END   ADRS ? &H",EA$
1060 '
1070 SA=VAL("&H"+SA$) AND &HFFF0
1080 EA=VAL("&H"+EA$)
1090 '
1100 INPUT "PRINTER (p) or CRT (c) ";OT$
1110 IF OT$="p" THEN LP.F=1 ELSE IF OT$="c" THEN LP.F=0 ELSE 1100
1120 '
2000 FOR I=SA TO EA STEP 16
2010  IF LP.F=1 THEN LPRINT RIGHT$("000"+HEX$(I),4)" : ";
                 ELSE PRINT RIGHT$("000"+HEX$(I),4)" : ";
2020  FOR J=0 TO 15
2030    IF LP.F=1 THEN LPRINT RIGHT$("0"+HEX$(PEEK(I+J)),2)" ";
                  ELSE PRINT RIGHT$("0"+HEX$(PEEK(I+J)),2)" ";
2040  NEXT J
2050  IF LP.F=1 THEN LPRINT ELSE PRINT
2060 NEXT I
```

ここでは、PRINT文とLPRINT文を同じように扱うためのテクニックを紹介していきます。

8-2-1 CRTとプリンタへの出力をファイルとして扱う

N_{88}-BASICでは、CRT、プリンタ、カセット・ディスクなどの入出力装置に対して、ファイルという概念を用いています。(詳しくは第5章を見て下さい)。

そこで、PRINT文、LPRINT文の出力をファイルに対して行うという考え方を使ってみましょう。

ファイルを使うことによって、PRINT、LPRINT文は、PRINT# [ファイル番号] 文で置きかえることができます。

しかし、単なる置き換えだけでは動きません。どこに対して出力するかをあらかじめ指定しておく必要があります。

CRTへ出力する場合

OPEN "SCRN：" FOR OUTPUT AS#1

プリンタへ出力する場合

OPEN "LPT：" FOR OUTPUT AS#1

処理が終ったら、CLOSE文を実行しておくことを忘れないで下さい。

この方法で前のプログラムを書き直してみましょう。

```
1000 ' ----- MEMORY DUMP ----- II
1010 WIDTH 80,25
1020 DEFINT A-Z
1030 '
1040 INPUT "START ADRS ? &H",SA$
1050 INPUT "END   ADRS ? &H",EA$
1060 '
1070 SA=VAL("&H"+SA$) AND &HFFF0
1080 EA=VAL("&H"+EA$)
1090 '
1100 INPUT "PRINTER (p) or CRT (c) ";OT$
1110 IF OT$="p" THEN F$="LPT1:" ELSE IF OT$="c" THEN F$="SCRN:" ELSE 1100
1120 '
1130 OPEN F$ FOR OUTPUT AS #1
1140 '
2000 FOR I=SA TO EA STEP 16
2010  PRINT #1,RIGHT$("000"+HEX$(I),4)" : ";
2020  FOR J=0 TO 15
2030   PRINT #1,RIGHT$("0"+HEX$(PEEK(I+J)),2)" ";
2040  NEXT J
2050  PRINT #1,""
2060 NEXT I
2070 '
2080 CLOSE
```

これでいくらか無駄が省けました。

また別の方法として、CRTとプリンタに2つのファイルをオープンしておいて、ファイル番号を変数として使うことも考えられます。

```
1000 ' ----- MEMORY DUMP ----- II
1010 WIDTH 80,25
1020 DEFINT A-Z
1030 '
1035 OPEN "LPT1:" FOR OUTPUT AS #1
1036 OPEN "SCRN:" FOR OUTPUT AS #2
1037 '
1040 INPUT "START ADRS ? &H",SA$
1050 INPUT "END   ADRS ? &H",EA$
1060 '
1070 SA=VAL("&H"+SA$) AND &HFFF0
1080 EA=VAL("&H"+EA$)
1090 '
1100 INPUT "PRINTER (p) or CRT (c) ";OT$
1110 IF OT$="p" THEN F=1 ELSE IF OT$="c" THEN F=2 ELSE 1100
1120 '
2000 FOR I=SA TO EA STEP 16
2010  PRINT #F,RIGHT$("000"+HEX$(I),4)" : ";
2020  FOR J=0 TO 15
2030   PRINT #F,RIGHT$("0"+HEX$(PEEK(I+J)),2)" ";
2040  NEXT J
2050  PRINT #F,""
2060 NEXT I
2070 '
2080 CLOSE
```

※ただし、これらのプログラムは、N_{88}-DISK-BASICでのみ動作します。

8-2-2 PRINT to LPRINTコマンドを作る

新しく作るプログラムに対しては、上のテクニックを使うとよいことがわかりました。それでは、今までに作っていたPRINT文を使ったプログラムはどうしますか。何とかしてみたいところです。

そこで、次の実験をしてみましょう。

```
100 POKE &HE64C,1 : PRINT "ABC"
```

上のプログラムを実行すると、文字が画面ではなくプリンタに出力されます。

POKE文を実行しただけですが、このE64CH番地というのは何でしょう。実はこれは、1文字出力をどこに対して行うかを表わすワークエリアの番地なのです。ここの内容が0であれば画面、そうでなければプリンタに出力されます。

ここで、「なぁーんだ、それじゃプリンタに出したいときはPOKE &HE64C,1をやるだけでいいんだな。」と思ってしまってはいけないのです。次のプログラムを実行してみて下さい。

```
100 POKE &HE64C,1 : PRINT "ABC"
110 PRINT "DEF"
```

110行では画面に出力されますね。これは、E64CHというワークエリアには、1度プリント文が実行された後は、0が書き込まれてしまうためです。

そこで、次のプログラムを実行してみて下さい。

```
100 ' [[[ PRINT TO LPRINT ]]]
110 FOR I=&HF260 TO &HF286
120  READ D$ : POKE I,VAL("&H"+D$)
130 NEXT I
140 '
150 CH=&HEEB6:POKE CH,&HC3:POKE CH+1,&H60:POKE CH+2,&HF2
160 PH=&HEDCF:POKE PH,&HC3:POKE PH+1,&H7A:POKE PH+2,&HF2
170 '
180 DATA FE,95,28,0C,FE,EE,C2,93,03,D7,C2,93,03,AF,18,06
190 DATA D7,C2,93,03,3E,01,32,86,F2,C9,F5,3A,86,F2,B7,28
200 DATA 03,32,4C,E6,F1,C9,00
```

何も変化はありませんが、あなたのN_{88}-BASICにはCMDという便利なコマンドが追加されたわけです。

試しに次のようにやってみて下さい。

```
cmd on
Ok
print "ABCDEF"
Ok
```

PRINT文でプリンタへ文字が出力されましたね。うまく動かなかったら上のプログラムをもう一度確かめて実行して下さい。

次にもとに戻します。

```
cmd off
Ok
print "ABCDEF"
ABCDEF
Ok
```

今度は、画面に出力されますね。

さあ準備は整いました。前のプログラムをこのコマンドを使って書き直してみましょう。

```
1000 ' ----- MEMORY DUMP -----
1010 WIDTH 80,25
1020 DEFINT A-Z
1030 '
1040 INPUT "START ADRS ? &H",SA$
1050 INPUT "END   ADRS ? &H",EA$
1060 '
1070 SA=VAL("&H"+SA$) AND &HFFF0
1080 EA=VAL("&H"+EA$)
1090 '
1100 INPUT "PRINTER (p) or CRT (c) ";OT$
1110 IF OT$="p" THEN CMD ON ELSE IF OT$="c" THEN CMD OFF ELSE 1100
1120 '
2000 FOR I=SA TO EA STEP 16
2010  PRINT RIGHT$("000"+HEX$(I),4)" : ";
2020  FOR J=0 TO 15
2030   PRINT RIGHT$("0"+HEX$(PEEK(I+J)),2)" ";
2040  NEXT J
2050  PRINT
2060 NEXT I
```

8-3. 漢字プリンタ(PC-8822)

8-3-1 使って便利な漢字←→キャラクタ対応表

漢字の出力には、漢字JISコード表を用います。これは、１つの漢字に対して２バイトの16進コードを与えるものです。

この表を使って、漢字列を印字するプログラムは次のようになります。

```
100 ' --- PC-8822 print kanji
110 '
120 KJ$=CHR$(27)+"K"
130 '
140 LPRINT KJ$;
150 FOR I=1 TO 8
160   READ HD,LD
170   LPRINT CHR$(HD);CHR$(LD);
180 NEXT
190 '
200 DATA &H46,&H7c,&H4b,&H5c,&H45,&H45,&H35,&H24
210 DATA &H33,&H74,&H3c,&H30,&H32,&H71,&H3c,&H52
日本電気株式会社
```

ところがこれは、次のようにしても良いわけです。

```
100 ' --- PC-8822 print kanji
110 '
120 KJ$=CHR$(27)+"K"
130 '
140 LPRINT KJ$;"F|K¥EE5$3t<02q<R"
150 '
日本電気株式会社
```

これは、漢字JISコードをキャラクタ２文字に置きかえて直接LPRINT文中に入れたものですが、こうするとプログラムも短くなるし、データ文として持ったときも⅕になります。ただ難点としては、漢字JISコードからキャラクタへ変換する手間が必要です。

そこで、この手間を省くために、漢字からキャラクタを得る一覧表を紹介します。

この表は次のようなもので、

```
[ ア ] -------------------------------------------------------
       亜=0!  唖=0"  娃=0#  阿=0$  哀=0%  愛=0&  挨=0'  姶=0(  逢=0)  葵=0*
       茜=0+  穐=0,  悪=0-  握=0.  渥=0/  旭=00  葦=01  芦=02  鯵=03  梓=04
       圧=05  斡=06  扱=07  宛=08  姐=09  虻=0:  飴=0;  絢=0<  綾=0=  鮎=0>
       或=0?  粟=0@  袷=0A  安=0B  庵=0C  按=0D  暗=0E  案=0F  闇=0G  鞍=0H
       杏=0I
[ イ ] -------------------------------------------------------
       以=0J  伊=0K  位=0L  依=0M  偉=0N  囲=0O  夷=0P  委=0Q  威=0R  尉=0S
       惟=0T  意=0U  慰=0V  易=0W  椅=0X                =0[  移=0¥  維=0]
       緯=0^  胃=0_  萎=0⊥  衣=0a  謂                          井=0f  亥=0g
         =0h  育=0i  郁=0j  磯=0
```

例えば、'愛'という漢字に対しては、'0&'という文字列が得られます。複数の漢字については、上の例のように、文字列を継いでLPRINTしてやればよいわけです。

ただし、この表にも欠点があります。キャラクターコード表を見てもわかるように、60Hのところはブランクになっていますね。この表では、20Hと区別するために、

60H='⊥'

としてありますが、この60Hはキーボードから入力することができません。

LPRINT"7";CHR$(&H60) (「劇」という文字を出力)

というようにしてもよいのですが、ファンクションキーを使うともっとスマートになります。

KEY 1,CHR$(&H60)

として、ファンクションキー1を押しみて下さい。空白が出力されますね。これが、キャラクターコード60Hの文字です。

これで、60H(表では'⊥')も入力できるようになりました。

この表は、付録のところにもありますが、自分なりにアレンジした表を作りたいという人のために、この表を出力するプログラムをあげておきます。

• 非漢字←→キャラクタ

```
1000 ' ----- PC-8822 ｷｺﾞｳ ｷｬﾗｸﾀ ﾋｮｳ -----
1010 '
1020 DEFINT A-Z
1030 '
1040 LPRINT CHR$(27)"H";
1050 LC=0
1060 LPRINT "｢ ｷｺﾞｳ ｣ "STRING$(52,"-"):LPRINT SPC(6);
1070 FOR I=&H2121 TO &H2770
1080  CH=I¥256 : CL=I MOD 256
1090  IF CL=&H7F THEN I=(CH+1)*256+&H21 : GOTO 1080
1100  LPRINT CHR$(27)"K"CHR$(CH)CHR$(CL);
1110  LPRINT CHR$(27)"H="CHR$(CH);
1120  IF CL=&H60 THEN CL=ASC("┴")
1130  LPRINT CHR$(CL)" ";
1140  LC=LC+1
1150  IF LC=10 THEN LC=0:LPRINT : LPRINT SPC(6);
1160 NEXT
```

• 漢字←→キャラクタ

```
1000 ' ----- PC-8822 ｶﾝｼﾞ ｷｬﾗｸﾀ ﾋｮｳ -----
1010 '
1020 DEFINT A-Z
1030 '
1040 LPRINT CHR$(27)"H";
1050 CCE=&H3021
1060 FOR HR=ASC("ｱ") TO ASC("ﾜ")
1070  CCS=CCE
1080  READ CCE
1090  LPRINT "｢ "CHR$(HR)" ｣ "STRING$(54,"-"):LPRINT SPC(6);
1100  LC=0
1110  FOR I=CCS TO CCE-1
1120   CH=I¥256 : CL=I MOD 256
1130   IF CL=&H7F THEN I=(CH+1)*256+&H21 : GOTO 1120
1140   LPRINT CHR$(27)"K"CHR$(CH)CHR$(CL);
1150   LPRINT CHR$(27)"H="CHR$(CH);
1160   IF CL=&H60 THEN CL=ASC("┴")
1170   LPRINT CHR$(CL)" ";
1180   LC=LC+1
1190   IF LC=10 THEN LC=0:LPRINT : LPRINT SPC(6);
1200  NEXT I
1210  LPRINT
1220 NEXT
1230 '
1240 DATA &H304a,&H3126,&H3141,&H3177
1250 DATA &H323c,&H346b,&H3665,&H3735,&H3843
1260 DATA &H3a33,&H3b45,&H3f5a,&H4024,&H4139
1270 DATA &H423e,&H434d,&H4445,&H4462,&H4546
1280 DATA &H4660,&H4673,&H4728,&H4729,&H4735
1290 DATA &H4743,&H485b,&H4954,&H4a3a,&H4a5d
1300 DATA &H4b60,&H4c23,&H4c33,&H4c3d,&H4c4e
1310 DATA &H4c69,&H4c7b,&H4d3d
1320 DATA &H4d65,&H4d78,&H4e5c,&H4e61,&H4f24
1330 DATA &H4f41,&H4f54
```

8-3-2 外字データ作成プログラム

PC-8822には、外字機能があります。

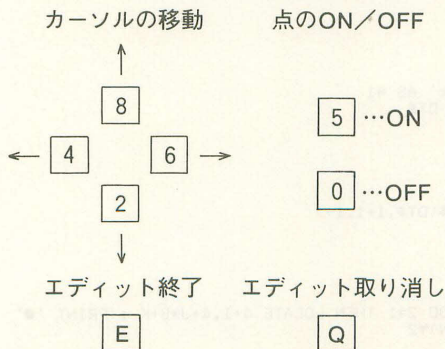
ここでは、その外字データを作成する簡単なエディタと、ディスクファイル上の外字データをPC-8822にロードするローダーを紹介します。

まず次のプログラムで、外字データファイルを初期化します。(外字データファイルは、ドライブ1のディスクに、GAIJI. datというファイル名で作られます。)

```
100 '
110 ' [[[ PC-8822 ｶﾞｲｼﾞ ﾃﾞｰﾀ ﾌｧｲﾙ ｲﾆｼｬﾗｲｽﾞ ]]]
120 '
130 GOSUB *OPEN.FILE
140 '
150 CONSOLE ,,1,0 : WIDTH 40,25
160 '
170 LOCATE 3,10
180 PRINT "PC-8822 ｶﾞｲｼﾞ ﾃﾞｰﾀ ﾌｧｲﾙ ｲﾆｼｬﾗｲｽﾞ"
190 LSET DT$=STRING$(32,0)
200 FOR I=1 TO 64
210   PUT #1,I
220 NEXT I
230 '
240 END
250 '
260 *OPEN.FILE
270  OPEN "GAIJI.dat" AS #1
280  FIELD #1,32 AS DT$
290 RETURN
```

この後、次のプログラムにより、任意の外字のエディットができます。

最初に外字コードを入力すると、(外字コードは、標準の64文字分としてあります) その外字のパターンが画面上に現われます。

テンキーを操作して、データを作成して下さい。キー操作は、次の通りです。

```
1000 '
1010 ' [[[ PC-8822 ｶﾞｲｼﾞ ﾃﾞｰﾀ ﾌｧｲﾙ ｻｸｾｲ ]]]
1020 '
1030 GOSUB *OPEN.FILE
1040 '
1050 CONSOLE ,,1,0 : WIDTH 40,25
1060 DEFINT A-Z
1070 DIM DT(31)
1080 '
1090 CLS
1100 LOCATE 0,10 : PRINT SPACE$(40)
1110 LOCATE 0,10 : INPUT "ｶﾞｲｼﾞ ｺｰﾄﾞ ( &H7620-&H765F )";G.CODE
1120 IF G.CODE<&H7620 OR G.CODE>&H765F THEN 1100
1130 '
1140 DT=G.CODE-&H761F
1150 '
1160 CLS
1170 PRINT "*** G.CODE : &H";HEX$(G.CODE);" ***"
1180 PRINT
1190 PRINT "    0123456789ABCDEF"
1200 PRINT
1210 FOR I=0 TO 15
1220  PRINT USING "  ! ················";HEX$(I)
1230 NEXT
1240 GOSUB *P.DATA
1250 '
1260 CX=4 : CY=4
1270 '
1280 LOCATE CX,CY
1290 KY$=INPUT$(1)
1300 ON INSTR("246850EeQq",KY$)
     GOSUB *DW,*LF,*RT,*UP,*ST,*RS,*EN,*EN,*QU,*QU
1310 GOTO 1280
1320 '
1330 *DW : IF CY<19 THEN CY=CY+1
1340  RETURN
1350 *LF : IF CX>4  THEN CX=CX-1
1360  RETURN
1370 *RT : IF CX<19 THEN CX=CX+1
1380  RETURN
1390 *UP : IF CY>4  THEN CY=CY-1
1400  RETURN
1410 *ST : PRINT "●";
1420  RETURN
1430 *RS : PRINT "·";
1440  RETURN
1450 *EN
1460  LOCATE 0,22 : PRINT "Sure ? (y/n)"; :S$=INPUT$(1)
1470  IF S$="n" THEN RETURN ELSE IF S$<>"y" THEN 1460
1480  LOCATE 0,22 : PRINT SPACE$(20);
1490  GOSUB *G.DATA
1500  LOCATE 0,22 : PRINT "Continue ? (y/n)"; : S$=INPUT$(1)
1510  IF S$="y" THEN 1090 ELSE IF S$<>"n" THEN 1500
1520 *QU
1530  CLS
1540 END
1550 '
1560 *OPEN.FILE
1570  OPEN "GAIJI.dat" AS #1
1580  FIELD #1,32 AS DT$
1590 RETURN
1600 '
1610 *P.DATA
1620  GET #1,DT
1630  FOR I=0 TO 31
1640   DT(I)=ASC(MID$(DT$,I+1,1))
1650  NEXT I
1660  FOR I=0 TO 15
1670   FOR J=0 TO 1
1680    CN=I*2+J
1690    FOR K=0 TO 7
1700     IF DT(CN) MOD 2=1 THEN LOCATE 4+I,4+J*8+K : PRINT "●"
1710     DT(CN)=DT(CN)¥2
1720    NEXT
```

```
1730   NEXT
1740  NEXT
1750 RETURN
1760 '
1770 *G.DATA
1780  FOR I=0 TO 15
1790   FOR J=0 TO 1
1800    CN=I*2+J
1810    AP=1
1820    FOR K=0 TO 7
1830     IF PEEK(&HF3C8+(4+J*8+K)*120+(4+I)*2)=ASC("●") THEN DT(CN)=DT(CN)+AP
1840     AP=AP+AP
1850    NEXT
1860   NEXT
1870  NEXT
1880  AP$=""
1890  FOR I=0 TO 31
1900   AP$=AP$+CHR$(DT(I))
1910  NEXT I
1920  LSET DT$=AP$
1930  PUT #1,DT
1940 RETURN
```

作成された外字データファイルは、次プログラムを作って、PC-8822にロードできます。

```
1000 '
1010 ' ｢｢｢ PC-8822 ｶﾞｲｼﾞ ﾃﾞｰﾀ ﾛｰﾄﾞ ｣｣｣
1020 '
1030 GOSUB *OPEN.FILE
1035 WIDTH LPRINT 255
1040 '
1050 CONSOLE ,,1,0 : WIDTH 40,25
1060 DEFINT A-Z
1070 '
1080 LOCATE 7,10
1090 PRINT "PC-8822 ｶﾞｲｼﾞ ﾃﾞｰﾀ ﾛｰﾄﾞ "
1100 '
1110 FOR I=1 TO 64
1120  GOSUB *P.DATA
1130 NEXT I
1140 '
1150 END
1160 '
1170 *OPEN.FILE
1180  OPEN "GAIJI.dat" AS #1
1190  FIELD #1,64 AS DT$
1200 RETURN
1210 '
1220 *P.DATA
1230  GET #1,I
1240  LOCATE 9,13
1250  PRINT "Loading .. &H";HEX$(&H761F+I)
1260  LPRINT CHR$(27)"*";
1270  LPRINT CHR$(&H76);CHR$(&H1F+I);
1280  FOR J=1 TO 32
1290   LPRINT MID$(DT$,J,1);
1300  NEXT
1310  LPRINT CHR$(4);
1320 RETURN
```

外字データファイルは、“GAIJI. dat”で統一されていますが、必要があれば、
*OPEN. FILEサブルーチンを変えて下さい。

8-4. WIDTH　LPRINTとTABコード

N$_{88}$-BASICでは、プリンタの印字桁数を制御するために、WIDTH　LPRINT文があります。
ここでは、WIDTH　LPRINT文のマニュアルにない使い方を見てみましょう。

8-4-1 WIDTH LPRINTの値と出力

WIDTH LPRINTの値と、その出力について実際に見てみましょう。(マニュアルとは違った動作をするところがありますので注意が必要です。)
動作チェックのために次のプログラムを用います。(プリンタは80ケタのものを使用)

```
100 INPUT LW
110 WIDTH LPRINT LW
120 FOR I=1 TO 300
130   LPRINT "*";
140 NEXT I
```

①LW　(WIDTH　LPPRINTの値)がプリンタの幅と同じかまたは小さい場合
(LW＝60)

```
************************************************************
************************************************************
************************************************************
************************************************************
************************************************************
```

この場合は、60文字印字したところで改行しています。

②LWがプリンタの幅より大きい場合(LW＝100)

```
********************************************************************************
********************
********************************************************************************
********************
********************************************************************************
********************
```

この場合は、80文字で改行（プリンタ側で行なわれる）した後、20文字めで再び改行しています。PC本体から見れば100文字めで改行したことになりますね。
普通こういった使い方はしないでしょう。

③LWが0の場合

マニュアルでは、LWが0のときは256と解釈されるとなっていますが、実際には、1文字印字して2回改行することの操り返しになります。

これも使える値ではないようですね。

```
*

*

*

•
•
•
```

④LWが255の場合

マニュアルにはないのですが、LWが255のときは特別の働きをします。一見②と同じようになると思えますが、実際は、WIDTH LPRINTの値は無視されることになります。つまり、改行はプリンタまかせということです。

このことは、不特定の桁数のプリンタ用のプログラムを作成するときに意味があります。

なお、リセット時には、この値は255となっています。同じプログラムでも、プリンタ出力が異なる現象が起こるのは、この値が違っている場合が多いようです。

プリンタ用のソフトには、念のために、WIDTH LPRINT 255を入れておくことをおすすめします。

8-4-2 水平タブコードの出力とドット対応グラフィック

WIDTH LPRINT 255にはもう１つの機能があります。

次のプログラムを実行すると、画面上では、TABコードが正常に出力されますが、プリンタでは、次のように異ったものが出力されます。

```
100 WIDTH LPRINT 255
110 TAB.CODE$=CHR$(9)
120 PRINT "ABC";TAB.CODE$;"DEF"
130 LPRINT "ABC";TAB.CODE$;"DEF"
```

プリンタ: ABCDEF

画　面: ABC　　　　DEF

ここでWIDTH LPRINTの値を80にすると、プリンタにも画面と同じ出力が得られますね。

これはどういうことかというと、N_{88}-BASICでは、TABコードの出力については、POS、LPOSの値を参照して、必要な数だけのCHR$(32)を出力するという方法をとっています。ところがWIDTH LPRINTの値が255のときは、このようなことはせずに、プリンタに対してCHR$(9)をそのまま送ります。そこでプリンタ側での水平タブ位置が設定されていない場合、上のように一見無視された形になるわけです。

逆に言えば、N_{88}-BASICでは、従来のようにめんどうな手順(PC-8821/22ユーザーズマニュアル３-５)使わなくても、水平タブコードを送ることが可能になったということです。

また、ドット対応グラフィックのデータ出力のときにも、次のような利点が出てきます。

N-BASICモードや、WIDTH　LPRINTの値が255以外のとき、次のプログラムを実行すると、CHR$(9)の出力に対して、３個のCHR$(32)がプリンタに送られ、Ⓐのような出力になります。

```
100 LPRINT CHR$(27)"S0256";
110 FOR I=0 TO 255
120   LPRINT CHR$(I);
130 NEXT
```

Ⓐ

これを避けるために、BASICのサブルーチンや、機械語によるサブルーチンを作っていたわけですが(PC-8821/22ユーザーズマニュアル3-19)WIDTH LPRINT 255によって、このような心配も不要になります。

実際WIDTH LPRINT 255を実行した後、上のプログラムを実行してみて下さい。目的とした出力Ⓑが得られます

Ⓑ

第9章　ディスク

第9章　ディスク

9-1. はじめに

この文章はN_{88}-DISK BASICのDISKに関する部分について述べてありますが、その前におことわりしておかなければならないことがあります。
①本書で述べるN_{88}-DISK BASICはシステム起動時に次の様に表示されるものです。

```
Disk version [Apr 24,1982]
How many files(0-15)?
```

日付がFeb 4,1982となっているものは古いバージョンです。また、Aug 10,1982となっているものは新しいバージョンです。
②両面ディスクの場合、表裏の２つのトラックを合わせてシリンダと呼ぶこともありますが、本章では使用していません。

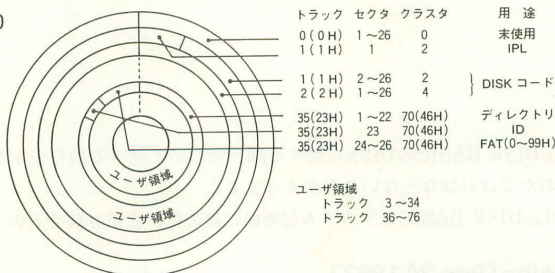
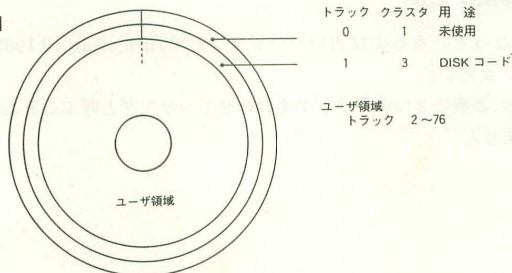
9-2. ディスクの構造

9-2-1 ディスク・マップ

①8インチ　両面

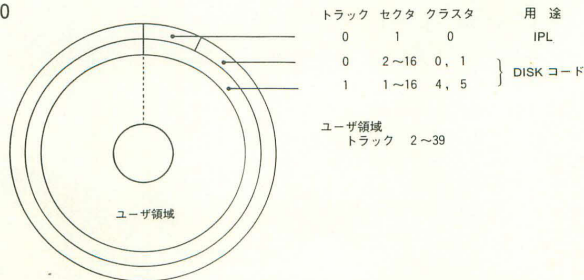
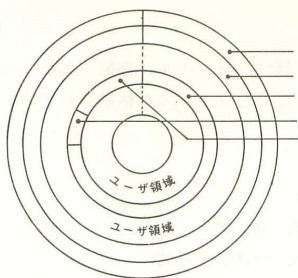
トラック数　片面77（使用しているのは片面76）

セクタ数　26セクタ／トラック

データ容量　1,011,712バイト（システム領域を含む）

②5インチ両面

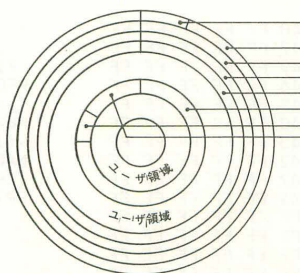
トラック数　片面40

セクタ数　16セクタ／トラック

データ容量　327,680バイト（システム領域を含む）

③5インチ片面

トラック数　35

セクタ数　16セクタ／トラック

データ容量　143360バイト（システム領域を含む）

9-2-2 ディスクアドレスとクラスタとの変換

1）5インチ片面のとき

〈クラスタ〉＝〈トラック〉×2＋〈セクタ〉￥9

〈トラック〉＝〈クラスタ〉￥2

〈セクタ〉＝（〈クラスタ〉MOD 2）×8＋1から8セクタ

2）5インチ両面のとき

〈クラスタ〉＝〈トラック〉×4＋〈サーフェス〉×2＋〈セクタ〉￥9

〈トラック〉＝〈クラスタ〉￥4

〈サーフェス〉＝（〈クラスタ〉MOD 4）￥2

〈セクタ〉＝（〈クラスタ〉MOD 2）×8＋1から8セクタ

3）8インチのとき

〈クラスタ〉＝〈トラック〉×2＋〈サーフェス〉

〈トラック〉＝〈クラスタ〉￥2

〈サーフェス〉＝〈クラスタ〉MOD 2

〈セクタ〉＝1セクタ～26セクタまで全部

9-2-3 ディレクトリ

ディレクトリには、ファイル名，ファイルの属性，ファイルが格納されている先頭クラスタ番号がしまわれています。これによりファイル名とファイルが格納されている場所との対応がつけられます。

ディレクトリの位置は、

5インチ片面…トラック18，セクタ1～12

5インチ両面…サーフェス1，トラック18，セクタ1～12

8インチ………サーフェス0，トラック35，セクタ1～22

となっています。1つのファイルに対して16バイトが割り当てられていますが、そのうち使用されているのは12バイトです。ディレクトリは図9-2-3のようになっています。

5インチ両面のとき

```
h]^d1,1,12,1
Track 12,Surface 01,Sector 01
0000 66 6F 72 6D 61 74 6E 38 38 80 48 FF FF FF FF FF    formatn88_H
0010 00 75 63 6B 75 70 6E 38 38 80 47 FF FF FF FF FF     uckupn88_G
0020 45 46 49 4C 45 53 6E 38 38 C0 49 FF FF FF FF FF    EFILESn88ﾀI
0030 46 45 44 49 54 20 6E 38 38 80 46 FF FF FF FF FF    FEDIT n88_F
0040 00 65 74 69 6E 66 6E 38 38 80 45 FF FF FF FF FF     etinfn88_E
0050 48 45 58 44 54 41 6E 38 38 90 4C FF FF FF FF FF    HEXDTAn88┴L
0060 78 66 69 6C 65 73 6E 38 38 80 45 FF FF FF FF FF    xfilesn88_E
0070 4D 41 43 48 4E 4C 62 69 6E 01 42 FF FF FF FF FF    MACHNLbin B
0080 54 43 48 4E 4F 57 64 74 61 00 47 FF FF FF FF FF    TCHNOWdta G
0090 50 52 54 45 43 54 6E 38 38 A0 41 FF FF FF FF FF    PRTECTn88 A
00A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
h]
```


（図9-2-3）

9-2-4 IDセクタ

IDにはユーザーズマニュアルにある通り、ディスク全体の属性、一度にOPENできるファイルの数，電源ON（リセット）時に実行される文が格納されています。

IDは

5インチ片面…トラック18，セクタ13

5インチ両面…サーフェス1，トラック18，セクタ13

8インチ………サーフェス0，トラック35，セクタ23

に割り当てられています。

```
hJ^d1,1,12,d（ミニ両面の場合）
Track 12,Surface 01,Sector 0D
0000 00 03 6B 65 79 20 32 2C 22 66 69 6C 65 73 20 22    key 2,"files "
0010 3A 70 72 69 6E 74 3A 66 69 6C 65 73 00 00 00 00    :print:files
0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

1度にOPENできるファイル数

属性（ディレクトリと同じ）

なお、1度にOPENできるファイル数と電源ON時に実行される文は、システムディスクでないと意味を持ちません。

9-2-5 FAT (File Allocation Table)

FATはファイルの格納状態を示します。ファイルが1クラスタに納まらない場合、残りを別のクラスタに書き込まなければなりません。この「別のクラスタ」をどこにするかにはいろいろな方法がありますが、N88-DISK BASICでは、適当に空いているクラスタに書き込みます。このとき、どこのクラスタに書き込んだかを記録しておかないとあとで困ることになります。また、「別のクラスタ」をさがす時に、どこが空きクラスタかが分からなければなりません。これらの情報を記録したものが、FATです。

ではこのFATはどのようになっているか見てみましょう。

FATの位置は

5インチ片面…トラック18，セクタ14～16，160バイト

5インチ両面…サーフェス1，トラック18，セクタ14～16，70バイト

8インチ………サーフェス0，トラック35，セクタ24～26，154バイト

となっていて、3つのセクタとも同じものが入っています。

h]^d1,1,12,e（5インチ両面の場合）

```
h]^d1,1,12,e
Track 12,Surface 01,Sector 0E
0000 FE FE FE FE FE FE FE FE FF FF FF FF FF FF FF FF ┐
0010 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF │
0020 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF │
0030 FF FF FF FF FF FF FF FF C2 FF FF 38 3B FF 3C C6 │                    ｯ  8; くｺ
0040 3F C1 40 3E C8 C8 44 43 C5 C8 FE FE 4D C1 FF FF │               ?ﾁ@>ﾈﾈDCﾅﾈ  Mﾁ
0050 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF │FAT
0060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF │160バイト
0070 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF │
0080 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF │
0090 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ┘
00A0 C6 87 E5 00 45 45 00 01 00 00 00 00 00 00 00 00 ┐                      ｺ▶ EE
00B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 │
00C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 │意味は
00D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 │ありません
00E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 │
00F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ┘
h]    +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
```

クラスタ47Hを見て下さい。これはTCHNOWdtaというファイルの先頭クラスタです。ここには43Hという値が入っています。これはデータがクラスタ47Hに入り切れず、クラスタ43Hに続いていることを示します。クラスタ43Hを見ると3EHとなっています。こうして

47H→43H→3EH→3CH→3BH→38H

と続いていきます。クラスタ38Hを見ると値はC2Hとなっています。クラスタC2Hというものはありません。値がC1H～C8H(8インチの時はC1H～DAH)の時はここのクラスタでファイルが終わっていて、下位5ビット(5インチの時1～8,8インチの時1～1AH)がそのクラスタで実際に使用しているセクタ数を表わします。ここではクラスタ38Hのうち2セクタを使用してファイルが終わっていることを示します。

これらをまとめると次のようになります。

バイトのデータ（16進）	クラスタの使用状態
8インチ両面の時 0～99 5インチ両面の時 0～9F 5インチ片面の時 0～45	使用中。連続したクラスタの一部であり、後続するクラスタを持つ。値が、後続するクラスタの番号を示している。
8インチ両面の時 C1～DA 5インチの時 C1～C8	使用中。連続したクラスタの最後のクラスタであり、下5(5インチの時は下4)ビットの内容が、そのクラスタで実際に使われているセクタの数を表わす。
FE	予約ずみのクラスタで、ファイルとして使うことはできない。(DISKコード、IPL、ディレクトリ、FAT自身を含むクラスタがこれである。)
FF	未使用

PC-8801 USER'S MANUALより
（表9-2-5）

9-3. ドライブテーブル

電源ON（リセット）時にドライブポインタとドライブテーブルが接続されているドライブの数だけ確保されます。各ドライブポインタ（２バイト）は、対応するドライブテーブルの先頭から３バイト目のアドレスを持っています。(図9-3-A)

ドライブテーブルの構造は図9-3-Bの通りです。

ドライブテーブルの先頭バイトがFFHの時はディスク上のFATがまだ更新されていませんので、そのドライブのディスクを抜いてはいけません。必ずCLOSEまたは、ENDを行なってから抜いて下さい。また、ディスクにファイルをOPENした時も必ずCLOSEまたはENDを行なってからでないとディスクを抜いてはいけません。これはN-DISK BASICのREMOVEに相当するのです。

(図9-3-B)

9-4. DSKF関数

DSKF関数は、ディスクファイルの構造に関する情報を返す関数で、機能を指定することによって、別表のような値が得られます。

この値のデータはDISK-BASIC上の8A18H番地から8A3BH番地に格納されています。

```
8A18 : 4C 1A 01 01 9A 23 1A 18 1A 03 17 34  : PC-8881
8A24 : 22 10 00 02 46 12 08 0E 10 03 0D 10  : PC-8031
8A30 : 27 10 01 02 A0 12 08 0E 10 03 0D 10  : PC-8031-2W
```

マニュアルでは、11種類の機能しかあげてありませんが、上のデータを見ると各ドライブタイプについて、12個ずつの値があるようです（DSKF機能一覧表）。

それからドライブタイプとしては、表の3つの他にDMA方式のミニフロッピーディスクドライブも考えられているようですが、これについてはPC-8031-2Wと同じようになっています。

DSKF機能一覧表

機能番号	PC-8881		PC8031-2W		PC-8031		機　能
	16進	10進	16進	10進	16進	10進	
0	4C	76	27	39	22	34	片面あたりの最大トラック番号
1	1A	26	10	16	10	16	1トラックあたりのセクタ数
2	01	1	01	1	00	0	片面…0, 両面…1
3	01	1	02	2	02	2	1トラックあたりのクラスタ数
4	9A	154	A0	160	46	70	クラスタの総数
5	23	35	12	18	12	18	ディレクトリが格納されているトラック番号
6	1A	26	08	8	08	8	1クラスタあたりのセクタ数
7	18	24	0E	14	0E	14	FATの開始セクタ番号
8	1A	26	10	16	10	16	FATの終了セクタ数
9	03	3	03	3	03	3	FATの個数
10	17	23	0D	13	0D	13	IDが格納されているセクタ番号
11*	34	52	10	16	10	16	

（＊N_{88}-DISK-BASICでは使われていません。）

9-5. 標準ディスク

9-5-1 物理的フォーマッティング

市販されている標準のフロッピーディスクを使用すると、NECのディスクよりも入出力が遅いことがあります。これはディスクの物理的フォーマットの方法が異なるためです。これをNECと同じ様にフォーマットしなおそうというものが物理的フォーマットプログラムです。これは機械語を使用しており、ディスクドライブコントローラであるμPD765を直接コントロールして物理的フォーマットを行ないます。

NECのディスク（PC-8886）と他社の多くのディスクとのフォーマットの違いは次のようなものです。一般的なディスクは、トラック上にセクタが1から26まで順番に並んでいま

す。ところがNECのディスクの場合にはセクタが1つおきに続いています。この形式をインタリーブ13と呼びます。第1セクタの次が第14セクタで、差が13だからです。なぜこのようなフォーマットにしたのかというと、PC-8881を使用して連続したセクタを読み書きする時にはこの方が速いからです。あるトラックの第1セクタ〜第3セクタを続けて読む場合を考えます。第1セクタを読みとった後、第2セクタを読みに行くまでの間にはいろいろな処理が必要です。このためセクタが順に並んでいる場合には、第2セクタを読みとろうとした時にはすでに第2セクタの先頭がヘッドを行きすぎてしまい、第2セクタがぐるっと1周してくるまで待たなくてはなりません。NECディスクの形式では第1セクタの次は第14セクタですから、第14セクタが行きすぎるまで待てばよいわけです。したがってこの形式の方がずっと速いわけです。もちろん1セクタだけのアクセスであれば、どちらの形式でも違いはありません。なお、このフォーマットの違いはあくまでも物理的（ハード的）なもので、論理的（ソフト的）にはまったく同じに扱えます。

```
100 '
110 ' STANDARD DISK PHYSICAL FORMAT
120 '
130 '   Copyright (C) 1982 by Radix
140 '
150 CLEAR,&HE3FF : DEFINT A-Z
160 CONSOLE ,,,0 : COLOR 0
170 PRINT "== PHYSICAL FORMAT OF STANDARD DISK =="
180 PRINT
190 PRINT "Mount ";:COLOR 4:PRINT " NEW ";:COLOR 0
200 PRINT " Diskette on Drive ";:COLOR 4:PRINT " 2 ":COLOR 0
210 PRINT "And hit return ";
220 IF INPUT$(1)<>CHR$(13) THEN 210 ELSE PRINT
230 INPUT "Sure (y/n) ";A$ : IF A$="n" THEN END
240 IF A$<>"y" THEN 230
250 '
260 PRINT "Working"
270 FOR I=&HE400 TO &HE510 : READ D : POKE I,D : NEXT
280 DEF USR=&HE400
290 FOR TR=1 TO 76
300  IF USR(TR)<>0 THEN *FAULT
310 NEXT
320 PRINT "Complete." : END
330 *FAULT
340  PRINT "DISK I/O FAULT in TRACK";TR : BEEP : END
350 '
360 DATA 126,50,246,228,62,8,211,245,62,0,50,245,228,33,17,229
370 DATA 6,26,17,247,228,58,245,228,15,15,79,58,246,228,119,35
380 DATA 113,35,26,119,35,62,1,119,35,19,16,239,62,7,50,62
390 DATA 239,62,15,205,148,60,58,245,228,246,1,205,148,60,58,246
400 DATA 228,205,148,60,205,113,60,62,0,50,61,239,50,22,239,58
410 DATA 38,239,230,224,254,32,194,228,228,58,39,239,71,58,246,228
420 DATA 184,194,231,228,62,17,211,98,62,229,211,98,62,103,211,99
430 DATA 62,128,211,99,62,166,211,104,62,2,211,243,58,246,228,254
440 DATA 38,62,15,56,2,62,63,211,244,62,255,50,62,239,62,77
450 DATA 205,148,60,58,245,228,246,1,205,148,60,62,1,205,148,60
460 DATA 62,26,205,148,60,62,54,205,148,60,62,64,205,148,60,205
470 DATA 113,60,62,165,211,104,62,0,211,243,205,247,58,58,38,239
480 DATA 71,58,245,228,246,1,184,194,234,228,58,39,239,183,194,237
490 DATA 228,58,245,228,183,62,4,50,245,228,202,13,228,33,0,0
500 DATA 34,65,236,201,46,1,1,46,2,1,46,3,1,46,4,38
510 DATA 0,34,65,236,201,0,0,1,14,2,15,3,16,4,17,5
520 DATA 18,6,19,7,20,8,21,9,22,10,23,11,24,12,25,13
530 DATA 26
```

PC-8881のドライブ２に新しい（またはデータを消されてよい）ディスクを入れます。そうしてこのプログラムをRUNすると、しばらくドライブ２の下のLEDがつきっぱなしになります。現在フォーマットしているわけです。LEDが消えると終わりです。しかしこのままではN$_{88}$-DISK-BASICのディスクとしては使えません。物理的にフォーマットしただけで、ディレクトリやFATの書き込みが行なわれていないからです。システムディスクにある「format.n88」を使ってソフト的なフォーマットを行なって下さい。いうなればこのプログラムはN-DISK-BASICのFORMAT文に相当するものです。

9-5-2 トラック０

標準ディスクの場合、N$_{88}$-DISK-BASICではトラック０が表裏とも使用されていません。これはなぜかというと、トラック０は他のトラックとは物理的なフォーマットが違っているからです。モニタで読んでみようとしてもまったく読めません。

トラック０のサーフェス０は単密度（FM方式，128バイト／セクタ）でフォーマットされています。そのため通常の方法では読み書きできません。ぜひとも読みたい場合は、μPD765を直接コントロールするプログラムを書く必要があります。

トラック０の内容は通常次のようになっています。(図9-5-2)

トラック０，サーフェス０

セクタ	
0 1	IPL用予備
0 2	IPL用予備
0 3	スクラッチ用予備
0 4	予　備
0 5	エラーマップ
0 6	予　備
0 7	ボリュームラベル
0 8	ファイルラベル１
0 9	ファイルラベル２
≈	≈
2 6	ファイルラベル19

(図9-5-2)

セクタ５：エラーマップ

フォーマット時に不良トラックの情報が書き込まれています。

セクタ７：ボリュームラベル

ボリューム名やユーザIDのほか、他のトラックのセクタ長やセクタの並んでいる順序などの情報が入っています。

セクタ８～26：ファイルラベル

ファイル名やその場所が書かれます。IBM方式のファイルのディレクトリです。

これらの情報は、ディスクをIBM方式で使用する時に必要なもので、EBCDICコードを使用していて、N$_{88}$-DISK-BASICではまったく使用されていません。前節のフォーマットのプログラムでもトラック０は考慮していません。第一、NECのディスク（PC-8886）は、トラック０，サーフェス０は単密度になっているものの、何も書き込まれていません。そのためもあって、ディスクの箱にはUNFORMATEDのシールが貼られています。

トラック０でもサーフェス１は倍密度で書かれているのですが、読めない場合があります。ですが強引に読むことはできます。モニタの^rコマンドを使うとエラーにはなりますが、メ

モリ上にはちゃんと読み込まれています。また、書き込みは自由です。いったん書き込むとその後は正常に読むことができます。

なお、前節の物理的フォーマットプログラムを変更して、トラック0からフォーマットするようにすると、トラック0もBASICで使用できるようになります（といってもDSKI$, DSKO$でしか使えませんが）。

9-6. BASICによるユーティリティ

9-6-1 拡張FILES

ファイル名とファイルの大きさだけでなく、その属性や、使用しているクラスタ番号、機械語ファイルの場合には、そのアドレスも出力する「拡張FILES」をBASICで作った例を紹介します。9-6-1のプログラムです。実行例を下に示します。

```
CRT (c) or PRINTER (p) ? c
DRIVE NUMBER ? 1

      DRIVE 1            FREE 133

FILE-NAME   ATTR  ADDRESS    LOCATION
format.n88   N   ----------   48(5)
EFILES.n88   R   ----------   49(8)
FEDIT  .n88  N   ----------   46 44(8)
HEXDTA.n88   P   ----------   4C 4D(1)
xfiles.n88   N   ----------   45(8)
MACHNL*bin   N   D000-E561    42 40 3F(6)
TCHNOW dta   N   ----------   47 43 3E 3C 3B 38(2)
PRTECT.n88   E   ----------   41(1)
Ok
```

プログラムでは、ディレクトリをDSKI$関数で読んでいくことにより、ファイル名，属性，先頭クラスタ番号を得ています。ファイルが使用しているクラスタ番号は、FATを読んで、順に追ってゆけばわかります。最後のカッコの中の数値は、最後のクラスタの中で実際に使用しているセクタ数を表わします。

機械語ファイルの場合はアドレスを読んできます。アドレスは、ファイル自身の最初の4バイトに開始番地，終了番地＋1が書き込まれています。

```
100 '
110 '    EXPANDED  FILES
120 '
130 '    Copyright (C) 1982 by Radix
140 '
150 DEFINT A-Z
160 PRINT : PRINT "CRT (c) or PRINTER (p) ? "; : E$=INPUT$(1) : PRINT E$
170 IF E$<>"c" AND E$<>"p" THEN 160
180 IF E$="c" THEN DEVICE$="scrn:" ELSE DEVICE$="lpt1:"
190 PRINT "DRIVE NUMBER ? "; : E$=INPUT$(1) : PRINT E$
200 DRIVE=VAL(E$) : IF DRIVE<1 OR DRIVE>PEEK(&HEC7D) THEN 190
210 OPEN DEVICE$ FOR OUTPUT AS#1
220 '
230 DIM FILE$(15),EX$(15),ATR$(15),CL$(15),FAT(DSKF(DRIVE,4)-1)
240 FOR I=0 TO 15
250   FIELD#0,I*16 AS DM$,6 AS FILE$(I),3 AS EX$(I),1 AS ATR$(I),1 AS CL$(I)
260 NEXT
270 DTYPE=3+(DSKF(DRIVE,1)=26)-DSKF(DRIVE,2)
280 IF DTYPE=1 THEN DM$=DSKI$(DRIVE,0,35,24) '8 inch
290 IF DTYPE=2 THEN DM$=DSKI$(DRIVE,1,18,14) '5 inch DS
300 IF DTYPE=3 THEN DM$=DSKI$(DRIVE,18,14)   '5 inch SS
310 FOR I=0 TO DSKF(DRIVE,4)-1
320   FAT(I)=ASC(MID$(DM$,I+1,1))
330   IF FAT(I)=255 THEN FREE=FREE+1
340 NEXT
350 '
360 PRINT
370 PRINT #1,"██████  DRIVE";DRIVE;" ██████";
380 PRINT #1,"     FREE";FREE
390 PRINT #1,
400 PRINT #1,"FILE-NAME  ATTR  ADDRESS   LOCATION"
410 FOR SECTOR=1 TO DSKF(DRIVE,10)-1
420   IF DTYPE=1 THEN DM$=DSKI$(DRIVE,0,35,SECTOR)
430   IF DTYPE=2 THEN DM$=DSKI$(DRIVE,1,18,SECTOR)
440   IF DTYPE=3 THEN DM$=DSKI$(DRIVE,18,SECTOR)
450   FOR I=0 TO 15
460     P=ASC(FILE$(I))
470     IF P=0   THEN *NEXT.FILE
480     IF P=255 THEN END
490     ATR=ASC(ATR$(I)) : ATR$=""
500     IF ATR AND &H80 THEN FTYPE$="." ELSE FTYPE$=" "
510     IF ATR AND 1    THEN FTYPE$="*"
520     IF ATR AND &H40 THEN ATR$="R"
530     IF ATR AND &H20 THEN ATR$=ATR$+"E"
540     IF ATR AND &H10 THEN ATR$="P"
550     IF ATR$="" THEN ATR$="N"
560     ADRS$=STRING$(9,"-")
570     IF FTYPE$="*" THEN GOSUB *GET.ADRS
580     PRINT #1,FILE$(I);FTYPE$;EX$(I);"  ";ATR$;"   ";ADRS$;"  ";
590     CL=ASC(CL$(I))
600     WHILE CL<&HC0
610       PRINT #1," ";RIGHT$("0"+HEX$(CL),2);
620       CL=FAT(CL)
630     WEND
640     PRINT #1,"(";MID$(STR$(CL-&HC0),2);")"
650   *NEXT.FILE
660   NEXT
670 NEXT : END
680 '
690 *GET.ADRS
700 OPEN MID$(STR$(DRIVE),2)+":"+FILE$(I)+EX$(I) FOR INPUT AS #2
710 ADRS$=RIGHT$("000"+HEX$(CVI(INPUT$(2,2))),4)+"-"
720 ADRS$=ADRS$+RIGHT$("000"+HEX$(CVI(INPUT$(2,2))-1),4)
730 CLOSE #2
740 RETURN
```

9-6-2 ディスクエディット

ディスケットをセクタ単位で書き換える場合には普通はモニタで次のようにします。

①^rでメモリ上に読み込む

②eでそのデータを修正する

③^wでディスクに書く

④^dコマンドで確かめる

しかし、^wで書く時に間違ったセクタに書き込んでしまったら大変です。またeコマンドでは16進または8進での修正しかできません。ここで紹介するプログラムは読み込んだセクタに書きこむことを原則とし、文字での修正も可能なディスクエディタです（プログラム9-6-2)。

RUNするとまず“Drive?”とたずねてきます。修正したいディスクのドライブ番号を入力して下さい。ここで、0と入力すると、ディスクからでなく、メモリーからデータを読むことになります。

次にサーフェス、トラック、セクタを順次入力して下さい。

Drive?〈ドライブ番号〉

1～のとき	0のとき
〈サーフェス入力〉	〈アドレス入力〉
〈トラック入力〉	
〈セクタ入力〉	

下のように表示されます。

```
■ DISK EDIT ■      Drive 1  Surface 1  Track 20  Sector 10

   +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
0: 28 4D 58 F3 0F 0B 2C 4D 59 29 2C 11 00 3C 09 58
1: 07 20 20 4D 58 F1 28 4D 41 52 4B 32 20 FE 20 0F
2: 10 29 20 F5 0F 10 20 3A 4D 59 F1 28 4D 41 52 4B
3: 32 20 FD 20 0F 10 29 20 F3 15 00 5A 09 62 07 20
4: 20 CB 20 40 28 4D 58 2C 4D 59 29 F4 28 4D 58 F3
5: 0F 0B 2C 4D 59 29 2C 11 00 8A 09 6C 07 20 20 4D
6: 58 F1 28 43 55 52 53 4F 52 20 FE 20 0F 10 29 20
7: F5 0F 10 20 3A 4D 59 F1 28 43 55 52 53 4F 52 20
8: FD 20 0F 10 29 20 F3 15 00 A8 09 76 07 20 20 CB
9: 20 40 28 4D 58 2C 4D 59 29 F4 28 4D 58 F3 0F 0B
A: 2C 4D 59 29 2C 11 00 B4 09 80 07 20 20 9D 20 2C
B: 2C 11 00 CD 09 8A 07 20 20 FF DB 20 12 2C 22 6C
C: 6F 61 64 22 F3 FF 96 28 0F 22 29 00 E5 09 94 07
D: 20 20 FF DB 20 16 2C 22 72 75 6E 22 F3 FF 96 28
E: 0F 0D 29 00 FF 09 9E 07 20 20 FF DB 20 0F 0A 2C
F: 22 63 6F 6E 74 22 F3 FF 96 28 0F 0D 29 00 0B 0A
```

```
EDIT,SAVE,FIN (e/s/f) ? ■
```

ここで次の3つを選びます。

EDIT　(e↵と入力)…データの修正
SAVE　(s↵　〃　)…データの書き込み
FIN　(f↵　〃　)…終わり

EDITモードでは3つのファンクションキーに意味があります。

F・1：16進で入力する時に押します。最初はこのモードです。
F・2：文字で入力する時に押します。カーソルが右の文字が表示されている領域に移動します。
F・5：EDITモードから抜け出します。

(プログラム9-6-2)

```
1000 '
1010 '      DISK EDITOR
1020 '
1030 '  Copytight (C) 1982 by Radix
1040 '
1050 '¥¥¥¥¥ INIT ¥¥¥¥¥
1060 DEFINT A-Z
1070 WIDTH 80,25 : CONSOLE 0,25,0,0 : COLOR 0
1080 KEY 1,CHR$(253)+"HEXCODE"+CHR$(29)
1090 KEY 2,CHR$(254)+CHR$(28)+"CHARACTER"
1100 KEY 5,CHR$(255)+"END"
1110 CLS : PRINT "[[ DISKETTE EDITOR ]]"
1120 INPUT "Drive    ";DRIVE : IF DRIVE=0 THEN 2230
1130 IF DSKF(DRIVE,1)=26 THEN DTYPE=3
1140 IF DSKF(DRIVE,2) THEN DTYPE=2 ELSE DTYPE=1
1150 IF DTYPE>1 THEN INPUT"Surface ";SURF
1160 INPUT "Track    ";TRACK
1170 INPUT "Sector   ";SECTOR
1180 FIELD#0,128 AS A$,128 AS B$ : GOSUB *DISK.READ
1190 '¥¥¥¥¥ DATA DISPLAY ¥¥¥¥¥
1200 CONSOLE ( 25 : CLS
1210 PRINT "███  DISK  EDITOR  ███    ";
1220 PRINT "Drive";DRIVE;
1230 IF DTYPE>1 THEN PRINT " Surface";SURF;
1240 PRINT " Track";TRACK;" Sector";SECTOR
1250 LOCATE 2,2
1260 FOR I=0 TO 15 : PRINT " +";HEX$(I); : NEXT
1270 PRINT SPC(6);"0123456789ABCDEF"
1280 LOCATE 3,3 : PRINT STRING$(47,"-");SPC(5);"┌";STRING$(16,"-");"┐"
1290 LOCATE 0,4
1300 FOR I=0 TO 15
1310    PRINT HEX$(I);": ";
1320    POKE &HF5DF+I*120,150
1330    FOR J=0 TO 15 : K=I*16+J
1340       IF I>7 THEN D=ASC(MID$(B$,K-127,1)) ELSE D=ASC(MID$(A$,K+1,1))
1350       PRINT RIGHT$("0"+HEX$(D),2);" ";
1360       POKE &HF5E0+I*120+J,D
1370    NEXT
1380    LOCATE 72,I+4 : PRINT"|"
1390 NEXT
1400 LOCATE 3,20 : PRINT STRING$(47,"-");SPC(5);"└";STRING$(16,"-");"┘"
1410 '¥¥¥¥¥  COMMAMD INPUT  ¥¥¥¥¥¥¥¥
1420 CONSOLE 22,1,0 : CLS
1430 INPUT "EDIT,SAVE,FIN (e/s/f) ";C$
1440 IF C$="" THEN 1420
1450 ON INSTR("esf",C$) GOSUB *EDITOR,*SAVER,*ENDER : GOTO 1420
1460 '¥¥¥¥¥ END ¥¥¥¥¥¥
1470 *ENDER
1480 CONSOLE 0,25,0 : RETURN 1490
1490 KEY 1,"load"+CHR$(34) : KEY 2,"files " : KEY 5,"run"+CHR$(13)
```

```
1500 LOCATE 0,22,1 : CONSOLE 0,25,1
1510 END : RUN
1520 '¥¥¥¥¥¥ SAVE ¥¥¥¥¥¥¥
1530 *SAVER
1540 WDRIVE=DRIVE : WSURF=SURF : WTRACK=TRACK : WSECTOR=SECTOR
1550 IF WDRIVE=0 THEN 1580
1560 INPUT "The same sector (y/n)";C$
1570 IF C$="y"OR C$="Y"THEN 1610
1580 INPUT "Drive ";WDRIVE
1590 IF DSKF(WDRIVE,2)=1 THEN INPUT "Surface,track,sector";WSURF,WTRACK,WSECTOR
1600 IF DSKF(WDRIVE,2)=0 THEN INPUT "Track,sector";WTRACK,WSECTOR
1610 '
1620 LOCATE ,,0:DIM D$(15):W=&HF5E0:H=VARPTR(W)
1630 FOR J=0 TO 15
1640   K=VARPTR(D$(J))
1650   POKE K,16:POKE K+1,PEEK(H):POKE K+2,PEEK(H+1)
1660   W=W+120
1670 NEXT
1680 A1$="" : FOR J=0 TO  7 : A1$=A1$+D$(J) : NEXT : LSET A$=A1$
1690 B1$="" : FOR J=8 TO 15 : B1$=B1$+D$(J) : NEXT : LSET B$=B1$
1700 IF DSKF(WDRIVE,1)=26 THEN DTYPE=3 : GOTO 1720
1710 IF DSKF(WDRIVE,2) THEN DTYPE=2 ELSE DTYPE=1
1720 GOSUB *DISK.WRITE : ERASE D$
1730 LOCATE ,,1 : RETURN
1740 '¥¥¥¥¥¥ EDIT ¥¥¥¥¥¥
1750 *EDITOR
1760 LOCATE 0,22 : PRINT"Now edit mode.";
1770 LOCATE 3,4 : CONSOLE ,,1
1780 C$=INPUT$(1) : V=ASC(C$)
1790 GOSUB *KEYCLEAR
1800 X=POS(0) : Y=CSRLIN
1810 IF V=255 THEN RETURN
1820 IF V=254 THEN IF X<52 THEN LOCATE X¥3+ 55,Y : GOTO 1780 ELSE 1780
1830 IF V=253 THEN IF X>52 THEN LOCATE X*3-165,Y : GOTO 1780 ELSE 1780
1840 IF X>52 THEN 2110
1850 '------ HEX EDIT ------
1860 IF C$>="0"AND C$<="9"THEN 1990
1870 IF C$>="A"AND C$<="F"THEN 1990
1880 IF C$>="a"AND C$<="f"THEN V=V-32:C$=CHR$(V):GOTO 1990
1890 IF V=8 THEN V=29
1900 IF V=32 THEN V=28
1910 IF V>=28 AND V<=31 THEN 1930
1920 GOTO 1780
1930 ON V-27 GOTO 1940,1950,1960,1970
1940 X=X+1-(X MOD 3>0)+(X=49)*48:IF X=3 THEN 1970 ELSE 1980
1950 X=X-1+(X MOD 3=0)-(X=3)*48:IF X=49 THEN 1960 ELSE 1980
1960 Y=Y-1-(Y=4) : GOTO 1980
1970 Y=Y+1+(Y=19)
1980 LOCATE X,Y:GOTO 1780
1990 PRINT C$;
2000 D=PEEK(X¥3+Y*120+&HF3FF)
2010 K=V-48+(V>60)*7
2020 IF X MOD 3=0 THEN D=(D AND &HF)OR(K*16)
2030 IF X MOD 3=1 THEN D=(D AND &HF0)OR K
2040 POKE X¥3+Y*120+&HF3FF,D
2050 IF X MOD 3=0 THEN 1780
2060 PRINT " ";
2070 IF X<49 THEN 1780
2080 IF Y=19 THEN LOCATE 3,4 ELSE LOCATE 3,Y+1
2090 GOTO 1780
2100 '----- CHARACTER EDIT -----
2110 IF V=8 THEN V=29
2120 IF V>=28 AND V<=31 THEN 2160
2130 POKE &HF3C8+Y*120+X,V
2140 LOCATE X*3-165,Y,0 : PRINT RIGHT$("0"+HEX$(V),2):LOCATE X,Y
2150 GOTO 2170
2160 ON V-27 GOTO 2170,2180,2190,2200
2170 X=X+1+(X=71)*16 : IF X=56 THEN 2200 ELSE 2210
2180 X=X-1-(X=56)*16 : IF X<71 THEN 2210
2190 Y=Y-1-(Y=4) : GOTO 2210
2200 Y=Y+1+(Y=19)
2210 LOCATE X,Y,1 : GOTO 1780
2220 '
2230 '¥¥¥¥  READ FROM MEMORY ¥¥¥¥
```

```
2240 PRINT"Read from memory.  Input address.";:INPUT ADRS$
2250 J=VAL("&H"+ADRS$) : A1$="" : B1$=""
2260 K=VARPTR(J) : D=VARPTR(A1$)
2270 POKE D,128 : POKE D+1,PEEK(K) : POKE D+2,PEEK(K+1)
2280 J=J+128 : D=VARPTR(B1$)
2290 POKE D,128 : POKE D+1,PEEK(K) : POKE D+2,PEEK(K+1)
2300 FIELD#0,128 AS A$,128 AS B$ : LSET A$=A1$ : LSET B$=B1$
2310 GOTO 1200
2320 '
2330 *DISK.READ
2340   IF DTYPE=1 THEN DUMY$=DSKI$(DRIVE,TRACK,SECTOR) : RETURN
2350   IF DTYPE=2 THEN DUMY$=DSKI$(DRIVE,SURF,TRACK,SECTOR) : RETURN
2360   IF DTYPE=3 THEN DUMY$=DSKI$(DRIVE,SURF,TRACK,SECTOR) : RETURN
2370   PRINT "DRIVE TYPE ERROR "; : BEEP : END
2380 *DISK.WRITE
2390   IF DTYPE=1 THEN DSKO$ WDRIVE,WTRACK,WSECTOR : RETURN
2400   IF DTYPE=2 THEN DSKO$ WDRIVE,WSURF,WTRACK,WSECTOR : RETURN
2410   IF DTYPE=3 THEN DSKO$ WDRIVE,WSURF,WTRACK,WSECTOR : RETURN
2420   PRINT "DRIVE TYPE ERROR "; : BEEP : END
2430 *KEYCLEAR
2440  STOP ON
2450   CKB=&H35D9 : CALL CKB 'clear queue
2460  STOP OFF
2470 RETURN
```

9-6-3 ファイルソート

Filesで出てくるファイル名は、ディレクトリに登録されている順番で表示されます。これをABC順にならべかえるプログラムを紹介しましょう。

RUNすると、ディレクトリを読み込んだ後、ソートします。ここで"MAY I WRITE ON DISK?"ときいてきますのでyまたはnを入力して下さい。yを入力すると、ソートしたディレクトリをディスクに書き込みます。

```
100 '
110 '   FILE  SORT
120 '
130 ' Copyright (C) 1982 by Radix  <H>
140 '
150 '--- TITLE
160 CONSOLE 0,25,1,0 : WIDTH 80,25
170 COLOR 0 : PRINT "N88-DISK BASIC [[  FILE SORT  ]] "
180 PRINT
190 '--- DRIVE NUMBER INPUT
200 INPUT "DRIVE NUMBER ";DRIVE
210 IF DSKF(DRIVE,1)=26 THEN DRIVE.TYPE=3:GOTO 240 'standard
220 IF DSKF(DRIVE,1)<>16 THEN *NOT.SUPPORT
230 IF DSKF(DRIVE,2) THEN DRIVE.TYPE=2 ELSE DRIVE.TYPE=1
240 PRINT
250 '--- READ DIRECTORY to FILE$( )
260 DIM FILE$(192),ROG$(15)
270 FOR I=0 TO 15 : FIELD#0, I*16 AS DUMY$,16 AS ROG$(I) : NEXT
280 SECTOR=1 : FILE.COUNT=0
290 IF DRIVE.TYPE=2 THEN DUMY$=DSKI$(DRIVE,1,18,SECTOR)
300 IF DRIVE.TYPE=1 THEN DUMY$=DSKI$(DRIVE,18,SECTOR)
310 IF DRIVE.TYPE=3 THEN DUMY$=DSKI$(DRIVE,0,DSKF(DRIVE,5),SECTOR)
320 FOR I=0 TO 15
330    IF ASC(ROG$(I))=255 THEN *FILEEND
340    IF ASC(ROG$(I))=0   THEN LSET ROG$(I)=STRING$(16,255)
350    FILE$(FILE.COUNT)=ROG$(I)
360    FILE.COUNT=FILE.COUNT+1
370 NEXT I
380 SECTOR=SECTOR+1 : IF SECTOR<DSKF(DRIVE,10) THEN 290
390 *FILEEND
400 FILE.COUNT=FILE.COUNT-1
410 '--- SORT
420 PRINT "---- SORTING ----"
430 FOR I=0 TO FILE.COUNT-1
440 FOR J=I+1 TO FILE.COUNT
450    IF FILE$(I)>FILE$(J) THEN SWAP FILE$(I),FILE$(J)
460 NEXT J,I
470 '--- WRITE ON DISK
480 INPUT"MAY I WRITE ON DISK (y/n) ";DUMY$
490 IF DUMY$<>"y" THEN PRINT "ABORTED." : END
500 J=0 : SECTOR=1
510 FOR I=0 TO FILE.COUNT
520    LSET ROG$(J)=FILE$(I)
530    J=J+1
540    IF J=16 THEN GOSUB *DISK.OUT : J=0 : SECTOR=SECTOR+1
550 NEXT I
560 IF J=0 THEN 610
570 WHILE J<16
580    LSET ROG$(J)=STRING$(16,255) : J=J+1
590 WEND
600 GOSUB *DISK.OUT
610 PRINT "WRITE END."
620 FILES DRIVE : END
630 '
640 *NOT.SUPPORT
650   PRINT "THAT DRIVE IS NOT SUPPORTED. ": BEEP
660   END
670 *DISK.OUT
```

```
680   IF DRIVE.TYPE=2 THEN DSKO$ DRIVE,1,18,SECTOR  : RETURN
690   IF DRIVE.TYPE=1 THEN DSKO$ DRIVE,18,SECTOR  : RETURN
700   IF DRIVE.TYPE=3 THEN DSKO$ DRIVE,0,DSKF(DRIVE,5),SECTOR  : RETURN
710   PRINT "DRIVE TYPE ERROR ": END
```

9-6-4 ファイル・リロケーション

今度はファイルをABC順ではなく、自分の好きな順番に並べ変えようというものです。このプログラムで処理できるのはディレクトリの最初の5セクタです。最大80個のファイルの並べ変えができます（プログラム9-6-4）。

RUNするとディレクトリを読み込んだ後、次のように表示されます。

```
DIRECTORY RELOCATION

SECTOR  1     SECTOR  2     SECTOR  3     SECTOR  4     SECTOR  5
------------------------------------------------------------------
Cpaint        #SORT  FAT    ( Empty )     ( Empty )     ( Empty )
DEMOUT        MEMDMP n88    ( Empty )     ( Empty )     ( Empty )
DEMO          SRCDSR        ( Empty )     ( Empty )     ( Empty )
Quartz        ( Killed )    ( Empty )     ( Empty )     ( Empty )
PATERN n88    ( Killed )    ( Empty )     ( Empty )     ( Empty )
SSMARK n88    CLSCNV n88    ( Empty )     ( Empty )     ( Empty )
SSLOGO n88    tpepo6 bin    ( Empty )     ( Empty )     ( Empty )
RLCDIR n88    ( Killed )    ( Empty )     ( Empty )     ( Empty )
SOTDIR n88    ( Killed )    ( Empty )     ( Empty )     ( Empty )
FEDIT  n88    ( Killed )    ( Empty )     ( Empty )     ( Empty )
HEXDTA n88    ( Killed )    ( Empty )     ( Empty )     ( Empty )
HEXBIN n88    ( Killed )    ( Empty )     ( Empty )     ( Empty )
backup n88    ( Killed )    ( Empty )     ( Empty )     ( Empty )
format n88    ( Killed )    ( Empty )     ( Empty )     ( Empty )
setinf n88    ( Killed )    ( Empty )     ( Empty )     ( Empty )
xfiles n88    ( Killed )    ( Empty )     ( Empty )     ( Empty )
------------------------------------------------------------------

  MARK        SWAP        RENAME                    END
```

(Killed)はkillされたファイルが入っていた所で、(Empty)はまだファイルが入っていない所です。右の文字が反転している場所はカーソルです。

カーソルキーとファンクションキーを使ってファイルを移動します。カーソルキーを押すと、その方向にカーソルが移動します。

MARK （f·1 キー）を押すとその場所がブリンクします。

SWAP （f·2 キー）を押すと、カーソルのあるファイルとマークされたファイル（ブリンクしている）とが入れかわります。

この MARK と SWAP でファイルの移動を行なっていくわけです。

RENAME （f·3 キー）は、カーソルのあるファイルの名前を変更するためのものです。

END （f·5 キー）を押すと次のようにたずねてきます。

"WRITE ON DISK?"　　(y/n/r)

yを入力…移動したデータをディスクに書き込みます

nを入力…ディスクに書き込まずに処理を終わります。(ファイルの位置は変わらなかったことになります)

rを入力…エディトモードに戻ります。

(プログラム9-6-4)

```
1000 '
1010 '   FILE  RELOCATION
1020 '
1030 '  Copyright (C) 1982 by Radix
1040 '
1050 CONSOLE 0,25,1,0 : WIDTH 80,25
1060 COLOR 0 : PRINT "N88-DISK BASIC [[  DIRECTORY RELOCATION ]] "
1070 PRINT
1080 DEFINT A-Z
1090 '--- DRIVE NUMBER INPUT
1100 INPUT "DRIVE NUMBER ";DRIVE
1110 IF DSKF(DRIVE,1)=26 THEN DRIVE.TYPE=3 : GOTO 1140
1120 IF DSKF(DRIVE,1)<> 16 THEN *NOT.SUPPORT
1130 IF DSKF(DRIVE,2) THEN DRIVE.TYPE=2 ELSE DRIVE.TYPE=1
1140 PRINT
1150 '--- READ DIRECTORY to FILE$( )
1160 DIM FILE$(79),ROG$(15)  'FILE# < 80
1170 FILE.END=0
1180 FOR I=0 TO 15 : FIELD#0, I*16 AS DUMY$,16 AS ROG$(I) : NEXT
1190 FOR SECTOR=1 TO 5
1200   IF DRIVE.TYPE=1 THEN DUMY$=DSKI$(DRIVE,18,SECTOR)
1210   IF DRIVE.TYPE=2 THEN DUMY$=DSKI$(DRIVE,1,18,SECTOR)
1220   IF DRIVE.TYPE=3 THEN DUMY$=DSKI$(DRIVE,0,DSKF(DRIVE,5),SECTOR)
1230   IF DRIVE.TYPE=0 THEN *NOT.SUPPORT
1240   FOR I=0 TO 15
1250     IF ASC(ROG$(I))=255 THEN FILE.END=1
1260     IF FILE.END=0 THEN FILE$(SECTOR*16-16+I)=ROG$(I)
1270     IF FILE.END=1 THEN FILE$(SECTOR*16-16+I)=STRING$(16,255)
1280   NEXT I
1290 NEXT SECTOR
1300 '--- DISPLAY
1310 CONSOLE 0,25,1,0 : WIDTH 80,25
1320 LOCATE 0,0 : PRINT "DIRECTORY RELOCATION"
1330 FOR SECTOR=1 TO 5
1340   LOCATE (SECTOR-1)*16,2 : PRINT " SECTOR  ";SECTOR
1350 NEXT
1360 LOCATE 0,3 : PRINT STRING$(79,"-")
1370 FOR I=0 TO 15
1380   FOR SECTOR=1 TO 5
1390     J=SECTOR-1
1400     LOCATE J*16,I+4
1410     FILE$=FILE$(J*16+I)
1420     GOSUB *MAKE.FILE.NAME
1430     PRINT " ";FILE$
1440   NEXT SECTOR
1450 NEXT I
1460 LOCATE 0,20 : PRINT STRING$(79,"-")
1470 '--- PREPARATION FOR EDIT
1480 CONSOLE ,,0
1490 KEY 1,CHR$(255)+"MARK"
1500 KEY 2,CHR$(&HFD)+"SWAP"
1510 KEY 3,CHR$(&HFC)+"RENAME"
1520 KEY 4,""
1530 KEY 5,CHR$(&HFE)+"END"
1540 CONSOLE ,,1
1550 COMMAND$=CHR$(31)+CHR$(30)+CHR$(29)+CHR$(28)
1560 COMMAND$=COMMAND$+CHR$(255)+CHR$(254)+CHR$(253)+CHR$(252)
1570 MARK=-1 : CURSOR=0
1580 '--- EDIT
1590 X=(CURSOR ¥ 16) *16 : Y=(CURSOR MOD 16) +4
1600 CCOLOR=4 : IF CURSOR=MARK THEN CCOLOR=6
1610 COLOR@ (X,Y)-(X+11,Y),CCOLOR
1620 LOCATE ,,0  : CKB=&H35D9 : CALL CKB
```

```
1630 P=INSTR(COMMAND$,INPUT$(1)) : IF P=0 THEN 1630
1640 ON P GOSUB *DN,*UP,*LF,*RI,*MARK,*EXIT,*SWAP.,*RENAME
1650 CCOLOR=0 : IF CURSOR=MARK THEN CCOLOR=2
1660 COLOR@ (X,Y)-(X+11,Y),CCOLOR
1670 CURSOR=NEWCURSOR
1680 GOTO 1590
1690 '
1700 *DN
1710   NEWCURSOR=CURSOR+1
1720   IF NEWCURSOR>=80 THEN NEWCURSOR=0
1730   RETURN
1740 *UP
1750   NEWCURSOR=CURSOR-1
1760   IF NEWCURSOR<0 THEN NEWCURSOR=79
1770   RETURN
1780 *LF
1790   NEWCURSOR=CURSOR-16
1800   IF NEWCURSOR<0 THEN NEWCURSOR=NEWCURSOR+80
1810   RETURN
1820 *RI
1830   NEWCURSOR=CURSOR+16
1840   IF NEWCURSOR>79 THEN NEWCURSOR=NEWCURSOR-80
1850   RETURN
1860 *MARK
1870   CKB=&H35D9:CALL CKB 'clear queue
1880   MX=(MARK ¥ 16) *16 : MY=(MARK MOD 16) +4
1890   IF MARK>-1 THEN COLOR@ (MX,MY)-(MX+11,MY),0
1900   MARK=CURSOR
1910   IF ASC(FILE$(MARK))=255 THEN BEEP : MARK=-1
1920   RETURN
1930 *EXIT
1940   CKB=&H35D9:CALL CKB 'clear queue
1950   MX=(MARK ¥ 16) *16 : MY=(MARK MOD 16) +4
1960   COLOR @(MX,MY)-(MX+11,MY),0
1970   MX=(CURSOR ¥ 16) *16 : MY=(CURSOR MOD 16) +4
1980   COLOR @(MX,MY)-(MX+11,MY),0
1990   CONSOLE ,,0
2000   KEY 1,"load "+CHR$(34)
2010   KEY 2,"files "
2020   KEY 3,"go to "
2030   KEY 4,"list "
2040   KEY 5,"run"+CHR$(13)
2050   CONSOLE ,,1
2060   LOCATE ,,1
2070   GOTO *WRITE.ON.DISK
2080 *SWAP.
2090   CKB=&H35D9:CALL CKB 'clear queue
2100   IF MARK<0 THEN BEEP : RETURN
2110   IF ASC(FILE$(CURSOR))=255 THEN BEEP : RETURN
2120   SWAP FILE$(MARK),FILE$(CURSOR)
2130   MX=(MARK ¥ 16) *16 : MY=(MARK MOD 16) +4
2140   FILE$=FILE$(MARK) : GOSUB *MAKE.FILE.NAME
2150   LOCATE MX,MY : COLOR 2 : PRINT " ";FILE$
2160   MX=(CURSOR ¥ 16) *16 : MY=(CURSOR MOD 16) +4
2170   FILE$=FILE$(CURSOR) : GOSUB *MAKE.FILE.NAME
2180   LOCATE MX,MY : COLOR 2 : PRINT " ";FILE$
2190   COLOR 0
2200   RETURN
2210 *RENAME
2220   CKB=&H35D9:CALL CKB 'clear queue
2230   P=ASC(FILE$(CURSOR)) : IF (P=0) OR (P=255) THEN BEEP : RETURN
2240   CONSOLE 22,23 : CLS : LOCATE,,1
2250   INPUT "NEW FILE NAME";FILE$ : IF FILE$="" THEN 2420
2260   IF (ASC(FILE$)=0) OR (ASC(FILE$)=255) THEN BEEP : GOTO 2250
2270   P=INSTR(FILE$,".") : NFILE$=STRING$(9," ")
2280   IF P=1 THEN BEEP : GOTO 2250
2290   IF P=0 AND LEN(FILE$)>6 THEN FILE$=LEFT$(FILE$,6)+"."+MID$(FILE$,7,3)
2300   IF P=0 THEN P=7
2310   MID$(NFILE$,1,6)=LEFT$(FILE$,P-1)
2320   MID$(NFILE$,7,3)=MID$(FILE$,P+1)
2330   FOR P=0 TO 79
2340     IF ASC(FILE$(P))=255 THEN P=79 : GOTO 2370
2350     IF LEFT$(FILE$(P),9)<>NFILE$ THEN 2370
2360     PRINT "EXIST FILE NAME.";CHR$(7) : GOTO 2250
```

```
2370   NEXT
2380   MID$(FILE$(CURSOR),1,9)=NFILE$
2390   MX=(CURSOR ¥ 16) *16 : MY=(CURSOR MOD 16) +4
2400   FILE$=FILE$(CURSOR) : GOSUB *MAKE.FILE.NAME
2410   LOCATE MX,MY : COLOR 2 : PRINT " ";FILE$
2420   COLOR 0 : LOCATE,,0
2430   CLS : CONSOLE 0,25
2440   RETURN
2450 '
2460 '
2470 *WRITE.ON.DISK
2480 LOCATE 0,22:INPUT "WRITE ON DISK (y/n/r) ";DUMY$
2490 IF DUMY$="y" THEN 2520
2500 IF DUMY$="r" THEN LOCATE 0,22 : PRINT SPC(30); : RETURN 1470 'EDIT
2510 PRINT "ABORTED." : END
2520 FOR SECTOR=1 TO 5
2530    FOR J=0 TO 15
2540       LSET ROG$(J)=FILE$(SECTOR*16-16+J)
2550    NEXT J
2560    GOSUB *DISK.OUT
2570 NEXT SECTOR
2580 FILES DRIVE
2590 END
2600 '
2610 '===== MAKE FILENAME ROUTINE =====
2620 *MAKE.FILE.NAME
2630 IF LEFT$(FILE$,9)=STRING$(9,255) THEN FILENAME$="( Empty ) " :GOTO 2660
2640 FILENAME$=LEFT$(FILE$,6)+" "+MID$(FILE$,7,3)
2650 IF ASC(FILENAME$)=0 THEN FILENAME$="( Killed )"
2660 FILE$=FILENAME$
2670 RETURN
2680 '
2690 '===== DSKO$ ROUTINE =====
2700 *DISK.OUT
2710   IF DRIVE.TYPE=1 THEN DSKO$ DRIVE,18,SECTOR : RETURN
2720   IF DRIVE.TYPE=2 THEN DSKO$ DRIVE,1,18,SECTOR : RETURN
2730   IF DRIVE.TYPE=3 THEN DSKO$ DRIVE,0,DSKF(DRIVE,5),SECTOR : RETURN
2740   PRINT "DISK TYPE ERROR ":END
2750 '
2760 '==== NOT SUPPORTED ERROR =====
2770 *NOT.SUPPORT
2780   PRINT "THAT DRIVE IS NOT SUPPORTED. ":END
```

第10章　RS-232C

第10章　RS-232C

10-1. RS-232C

RS-232Cとは、国際電信電話諮問委員会(CCITT)の勧告により、米国のEIAが決めた、標準シリアルインターフェイスのことです。これは、元来、データ端末装置（たとえばPC-8801）と、通信回線にデータを送受信するモデム（たとえば音響カプラ）とを接続する為に決められた規格です。ところが、この規格を使って、コンピュータ同士あるいは、コンピュータと周辺装置を接続する例が出てきたのです。現在では、１対１のシリアルインターフェイスとして、様々なシリアル入出力機器に用いられています。

図表10-1AにPC-8801に搭載されているRS-232Cのピン配置と各ピンの大まかな機能を示します。

端子番号	信号名	端子番号	信号名	ピンコネクション
1	GND	14	NC	
2	TXD	15	NC	
3	RXD	16	NC	
4	RTS	17	RXC	
5	CTS	18	NC	
6	DSR	19	NC	
7	GND	20	DTR	
8	DCD	21	NC	
9	NC	22	NC	
10	NC	23	NC	
11	NC	24	TXC	
12	NC	25	NC	
13	NC			

信号名	ピン番号	機　　能	信号名	ピン番号	機　　能
GND	1，7	1…保安用アース 7…信号用アース	DSR	6	モデムからの動作可能信号
TXD	2	送信データ	DCD	8	キャリア検出
RXD	3	受信データ	RXC	17	受信クロック入力
RTS	4	モデムへの送信要求	DTR	20	ターミナルの動作可能信号
CTS	5	モデムからの送信可能信号	TXC	24	送信クロック出力

（表10-1A）

10-1-1 モード指定

N_{88}-BASICで、このRS-232Cを使用するには、ファイルディスクリプタのデバイス名に、“com”を使用します。また、それに続いて、通信時のデータ形式や制御情報を指定し、いわゆるファイル名は存在しません。

制御パラメータのフォーマットと意味については、マニュアル16-4，16-5，16-12の項をご覧下さい。

例えば、偶数パリティ、データビット長8bit、ストップビット2bit、Xパラメータ有効、とすると、

“com : E83XN”

これを、ファイルディスクリプタとして使用します。なお、通信モードやスタック長の指定は、termコマンドの時のみ有効です。

PC-8801をターミナルモード専用機として使いたい時、背面にあるディップスイッチSW1,SW2を切り替え、電源ONと同時に、ターミナルモードにすることも可能で、各パラメータも、このスイッチで切り替えることができます。

(表10-1-1)

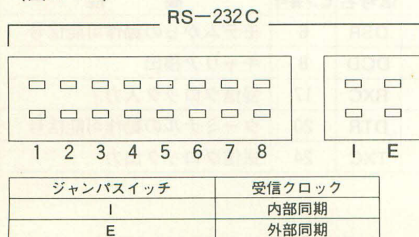
ディップスイッチ		ON(上向き)	OFF(下向き)
SW1	1	N-BASIC	N_{88}-BASIC
	2	ターミナルモード	BASICモード
	5	Sパラメータが有効	Sパラメータが無効
SW2	1	パリティチェック有り	パリティチェック無し
	2	偶数パリティ	奇数パリティ
	3	データ長が8ビット	データ長が7ビット
	4	ストップビットが2ビット	ストップビットが1ビット
	5	Xパラメータが有効	Xパラメータが無効
	6	半二重モード	全二重モード

10-1-2 ボーレイト

これでRS-232Cが使えるか？と言うと、まだです。シリアルインターフェイスを使う上で重要な事が、１つ残っています。それは、ボーレイト（データ転送の速度）の指定です。

ボーレイトの指定は、本体背面のジャンパスイッチで行なわれます。次の図で示す様にボーレイトは、８つの中から１つだけ選択し指定します。また、受信用クロックは、内部同期と外部同期の選択ができ、内部同期の場合は、先のジャンパスイッチの指定によるボーレイトとなります。

(図10-1-2)

ジャンパスイッチ	受信クロック
I	内部同期
E	外部同期

(表10-1-2A)

ジャンパースイッチ	ボーレート
1	75
2	150
3	300
4	600
5	1200
6	2400
7	4800
8	9600

(表10-1-2B)

10-2　コンピュータ同士を接なぐ

「PC-8801と他のコンピュータとの間で、データのやり取りを行いたい」、と言った事はよくあります。

例えば、会計、在庫管理システムにおいて、売り上げや出納、入出庫などのマスターファイルをホストコンピュータ上で作成、そのホストに多数のPCを接続し、各部課からのデータをPCに入力、ホストのファイルに登録する。あるいは、計算の一部をホストが行い、その間PCは他の計算を行う。そして結果をホストからもらい、PCの結果と合わせ最終結果を出力する、等々、数多くの応用が考えられます。

この様な時、ホスト側にRS-232Cポートがあれば、簡単にPCと接続できます。ただ、この時、注意しなくてはならないことがあります。

10-2-1 DTEとDCE

ある装置が、インターフェイスを介して通信回線に接続されている時、その装置をDTE（データ端末装置）、回線の終端装置であるインターフェイス部分をDCE（データ伝達装置）と言い、具体的にはそれぞれPC-8801と音響カプラなどです。

前に、RS-232Cはデータ端末装置とモデムを接続する為の規格だ、と述べましたが、言葉を変えると、これはDTEとDCEを接続する為の規格だ、とも言えます。

ここでちょっと困ったことが起こりました。コンピュータ同士を、RS-232Cで接続しようとすると、DTE同士の接続となってしまいます。DTE同士は、そのままでは接続できないのです。さて、それではどうするか。まず、DTEとDCEの違いを考えていくことにしましょう。次の表を見て下さい。

（表10-2-1）

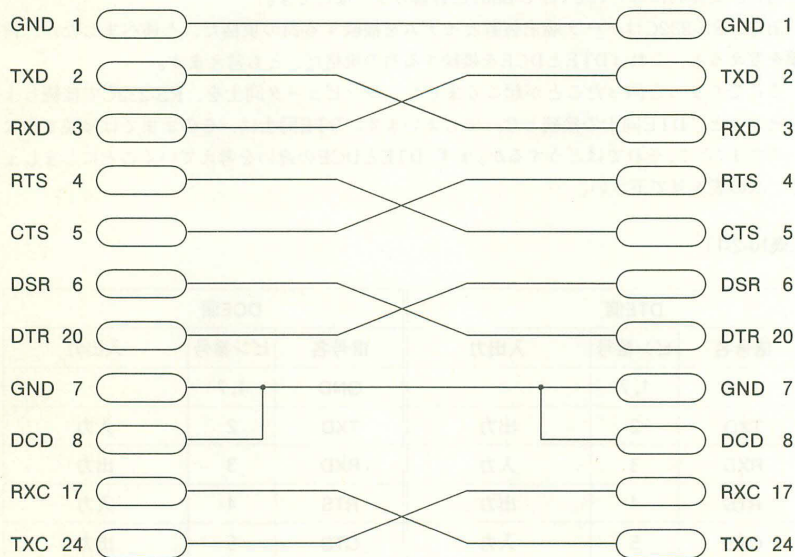
DTE側			DCE側		
信号名	ピン番号	入出力	信号名	ピン番号	入出力
GND	1,7	—	GND	1,7	—
TXD	2	出力	TXD	2	入力
RXD	3	入力	RXD	3	出力
RTS	4	出力	RTS	4	入力
CTS	5	入力	CTS	5	出力
DSR	6	入力	DSR	6	出力
DCD	8	入力	DCD	8	出力
RXC	17	入力	RXC	17	出力
DTR	20	出力	DTR	20	入力
TXC	24	出力	TXC	24	入力

各信号の入力と出力が全く逆です。また、実際の機械では、DTE側ではメスコネクタが、DCE側ではオスコネクタがついていて、一見しただけでどちらかわかります。しかもケーブルは、片方がオス、もう一方がメスコネクタとなっていて、DTE同士、DCE同士は接続できなくなっています。そこでちょっと細工をすることにしましょう。

10-2-2 専用ケーブルを作る

まず、RS-232C用オスコネクタを2つ用意します。次に、配線図の様にRxDとTxD、DSRとDTR、RTSとCTS、と言った対になっているピンを入れ替えて接続するのです。ただPCの場合、8番ピンのDCD（キャリア検出信号）は、対になるピンがなく、GND（7番ピン）に接続しておきます。これで、DCDは常にアクティブとなります。

ホストコンピュータによっては、PCにないピンが使用されていたり、逆にPCにあるピンが使用されていなかったり、あるいは、先のDCDに＋5～＋15Vの電圧をかける必要があるものもありますので、十分確認をして下さい。なお、この配線図はホストにPCを使うことを想定したものであり、結果的にPC-8801同士を接続する為のものです（図10-2-2)。

（図10-2-2）

これで、PC同士が接ながります。対になる信号を入れ替えたことで、各PCからは、他方が、等価的にモデムの様に見える為です。

使用しているピンの種類と、数が一致しているものならば、他の物も接続できます。この時、双方のボーレイトを合わせるのを、忘れないで下さい。

10-3. 2台のPCをつなぐ

前節で述べた専用ケーブルを使い、データのやり取りを行うには、2つの方法があります。

1つはtermモードによるもので、もう1つはBASICモードでのデータのやり取りです。

termモードでは、PCはインテリジェント・ターミナルとして働きます。〝インテリジェント〟というのは、単なるターミナルと違い、リモートBASICで、ある程度の処理を行わせることができる為で、ディスクを接続し、ディスクファイルを操作させる事も可能です。

一方、BASICモードでは、RS-232Cはファイルチャネルとして扱われ、プログラムファイル、データファイルの2種とも、やり取りが行えます。

ここでは、PC同士を接続することにして話を進めていきますが、データフォーマットを合わせれば、他の機械との接続でも同様に行えます。

10-3-1 データの転送

RS-232Cにデータを出力するには、他のシーケンシャル・ファイルと同様、次の様に行います。

```
10 OPEN "com:E83XS" FOR OUTPUT AS #1
20 A$="ABCDE,FGHIJ"
30 B$="KLMNO"
40 PRINT #1,A$,B$
50 CLOSE
```

この時、変数の間のコンマ“，”はデータの区切りではなく、画面の時と同様、タブ位置に右づめで合うように、スペースをつめる働きをします。この時のデータフォーマットは次の様になります。

（図10-3-1）

"A"	"B"	"C"	"D"	"E"	","	"F"	"G"	"H"	"I"	"J"	" "	" "	" "	"K"	"L"	"M"	"N"	"O"	CR	LF

画面に表示される時と、全く同じフォーマットです。これを次のプログラムで読むと、

```
10 OPEN "com:E83XS" FOR INPUT AS #1
20 INPUT #1,A$,B$
30 PRINT A$,B$
40 CLOSE
run
ABCDE          FGHIJ   KLMNO
Ok
```

読み込み時、“，”やCRコード(0DH)をデータの区切りと見なす為、データが途中でずれています。このことは逆に、データを区切る時は“，”を書き込まなくてはならないことを示しています。また、データ中に“，”を含める時は、ダブルクォーツ〃で囲みます。ダブルクォーツを使う時はCHR$(34)としなければなりません。

送信側

```
10 OPEN "com:E83XS" FOR OUTPUT AS #1
20 A$=CHR$(34)+"ABCDE,FGHIJ"+CHR$(34)
30 B$="KLMNO"
40 PRINT #1,A$;",";B$
50 CLOSE
```

受信側

```
10 OPEN "com:E83XS" FOR INPUT AS #1
20 INPUT #1,A$,B$
30 PRINT A$,B$
40 CLOSE
run
ABCDE,FGHIJ   KLMNO
Ok
```

LINE INPUTを使用すると、データの区切りがCR(0DH)のみとなりますので、“，”を含む場合でもダブルクォーツで囲む必要はありません。

さて、これまではディスクのシーケンシャルファイルと同様でしたが、1つだけそれと異って注意しなければならない事があります。それは、バッファの限界です。

RS-232Cの入力は、内部的には、インタラプトで処理されており、データを1文字受けるたびに、メインRAM上の、ファイルバッファに蓄え、INPUT文で、このバッファからデータを取り出すわけです。従って、INPUT文によるデータの取り出し速度より、受信したデータを蓄積していく速度の方が速ければ、バッファ内のデータはどんどん増え、ついにはバッファからあふれてしまいBO Errorとなってしまいます。特に、ボーレイトが速くデータの量が多い時は注意しなければなりません。

この現象を防ぐには、アクノリッジを返す方法があります。つまり、データを送ったら、相手が受け取った事を示す返事を出すまで、次の送出を待っているのです。その例を示します。

送信側

```
10 A$=CHR$(34)+"ABCDE,FGHIJ"+CHR$(34)
20 OPEN "com:E83XS" AS#1
30 FOR I=0 TO 5
40 PRINT #1,I;A$          :REM out data
50 INPUT #1,B$            :REM in acknowledge
60 NEXT I
```

受信側

```
10 OPEN "com:E83XS" AS #1
20 FOR I=0 TO 5
30 LINE INPUT #1,A$          :REM in data
40 PRINT #1,CHR$(4)          :REM out acknowledge
50 PRINT A$
60 NEXT I
```

アクノリッジの送出を待っているため、その分出力側の処理速度が落ちますが、10-4節の割り込みを使うと改善されます。

10-3-2 プログラムの転送

プログラムファイルを転送することも可能で、送信側でSAVE、受信側でLOADを行えばよいのです。ただ、この時受信側では、受信が完了してもLOADコマンドから抜け出さず、ころあいを見てSTOPキーを押さなくてはなりません。この現象について述べる前に、まず、プログラムファイル転送時のフォーマットについて考えましょう。

プログラムをRS-232CにSAVEするには、次の様に行います。

```
save "com:E83XS"
```

例えば、次のプログラムをSAVEしたとすると、転送時のフォーマットは、ディスクにアスキーセーブする時と同じで、次の様になります。

```
10 FOR N=0 TO 100:NEXT N
```

(図10-3-2)

“1”	“0”	“ ”	“F”	“O”	“R”	“ ”	“N”	“=”	“0”	“ ”	“T”	“O”	“ ”	“1”	“0”	“0”	“:”	“N”	“E”	“X”	“T”	“ ”	“N”	CR	LF

ただ1つ、ディスクのアスキー形式ファイルと違うのは、プログラムの最後を示すエンドマークがないことです。

LOADの時にはRS-232Cからの入力に対して、キー入力と全く同様の動作でプログラムを格納していきます。この時、別にエンドマークのたぐいは識別せず、また仮に識別していても、送信側で送らないのですから、いつまでたってもLOADコマンドから抜け出ません。

送信側と受信側がすぐ近くにあれば、SAVEコマンドの終了の後、受信側のSTOPキーを押せば、プログラムは正常に入っているはずです。が、送信側と受信側が離れている時などSAVEコマンドの終了を確認できない時どうすればよいでしょう。転送にかかる時間をあらかじめ調べておき、ころ合いを見てSTOPキーを押すのも1つの方法です。しかし、もっとよい方法があります。送信側で次に示す様に、ダイレクトモードで実行させるのです。

```
save "com:E83XS"
Ok
open "com:E83XS" for output as #1
Ok
print #1,"beep"
Ok
close
Ok
```

受信側では単にLOADコマンドのみで結構です。

```
load "com:E83XS"
?DS Error
Ok
```

LOADコマンドからエラーで抜け出していますが、プログラムは正常に入っています。なぜエラーとなるか分ると思いますが、送信側でプログラム・ファイルを送った後、ファイルをOPENして“beep”と送っています。これは、実は行番号をつけていなければ、何でもよいのですが、受信側ではこれをダイレクトステートメントと解し、DS Errorとなったわけです。これで、何とかプログラムを送ることができました。

ではダイレクトモードでのSAVEとしてプログラムを送るのでなく、BASICの管理下で、プログラムの送出ができないでしょうか？これが行なえるのです。次のプログラムを送信側で走らせ、受信側でLOADを実行させます。すると、

送信側

```
10 OPEN "com:E83XS" FOR OUTPUT AS #1
20 PRINT #1,"10 for i=0 to 100"
30 PRINT #1,"20 beep 1 : beep 0"
40 PRINT #1,"30 next i"
50 PRINT #1,CHR$(4)
60 CLOSE
```

受信側

```
load "com:E83XS"
?DS Error
Ok
list
10 FOR I=0 TO 100
20 BEEP 1 : BEEP 0
30 NEXT I
Ok
```

ちゃんと入ったでしょう。今まで、データファイルとプログラム・ファイルを別のものとして考えてきましたが、その差は行の初めに行番号があるかないかの違いのみで、本質的には同じものなのです。

次のプログラムは、ディスク上のアスキー形式プログラム・ファイルを、RS-232Cに送出するもので、受信側ではLOADを実行させます。

```
10 FILES : PRINT
20 INPUT "Type in file name ";NA$
30 IF LEN(NA$)>9 THEN PRINT" Too long ":GOTO 20
40 OPEN "1:"+NA$ FOR INPUT AS #1
50 OPEN "COM:E83XS" FOR OUTPUT AS #2
60 WHILE EOF(1)=0
70  LINE INPUT #1,A$
80  PRINT #2,A$
90 WEND
100 PRINT #2,CHR$(4)
110 CLOSE
```

逆に、送信側でSAVEで送出されたプログラムファイルを、データとして読むことも可能です。但し、この時、エンドマークがないので、あらかじめ何行のプログラムか、知る必要があります。次のプログラムは、送信側から送出されたプログラム・ファイル（先のプログラムによるもの）を受信し、受信側のディスク上に、アスキー形式プログラム・ファイルを作成するものです。

```
10 FILES:PRINT
20 INPUT "Type in files name",NA$
30 IF LEN(NA$)>9 THEN PRINT "Too long ": GOTO 20
40 OPEN "COM:E83XS" FOR INPUT AS #1
50 OPEN "1:"+NA$ FOR OUTPUT AS #2
60  LINE INPUT #1,A$
70  IF A$=CHR$(4) THEN 100
80  PRINT #2,A$
90  GOTO 60
100 CLOSE
```

10.4. RS-232Cによる割込み

N_{88}-BASICでは、RS-232Cでデータを受信した時、あらかじめ定義しておいたサブルーチンへ制御を移す機能、すなわち、割込み制御機能があります。

この機能を使うと、RS-232CからのINPUTの時、データが送られて来るまで何もせず待っている…と言った事がなくなるばかりでなく、送られて来たデータに対する応答が即座にできる、と言った利点があります。

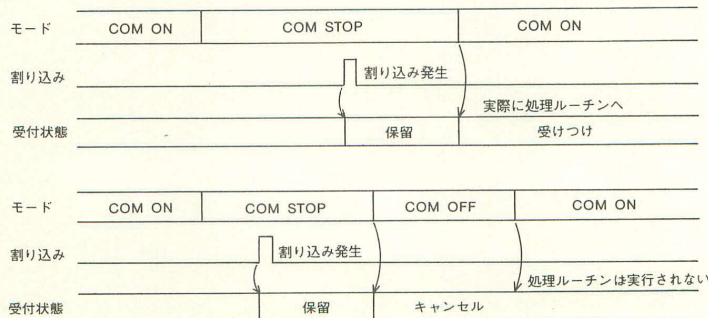
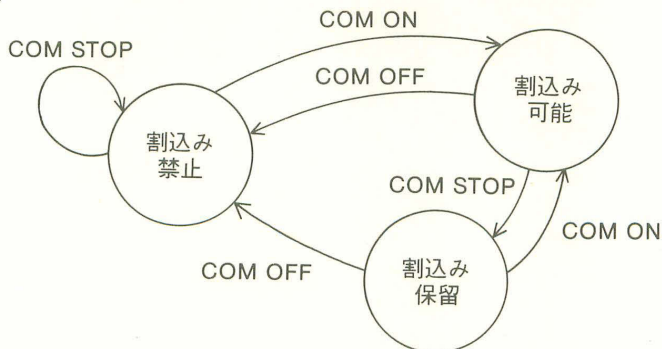
しかし、割込み処理を行うと言うことは、見かけ上割り込みサービスルーチンとメインルーチンが同時に走る為、一種のマルチタスク処理となり、各ルーチンの同期や排他、通信等つきつめれば難しい問題もあります。

ここでは、その使い方とキューバッファによる割込みルーチンとメインルーチン間のデータの送受信等を示します。

10-4-1 COM OFFとCOM STOP

プログラムの最初にON COM GOSUB ×××で割込み処理ルーチンを定義し、割込みを可能にするCOM ON命令を実行すれば、データを受信した時に、定義した処理ルーチンをコールします。この割込み関係のステートメントを示します。

ON COM GOSUB	割り込み処理ルーチンの定義
COM ON	割り込み可能
COM OFF	割り込み禁止
COM STOP	割り込み一時保留

この中で、COM OFFとCOM STOPの違いは明確にしておいて下さい。COM OFFは割込みそのものを無視するもので、COM OFFの間にデータが受信されても割り込みは起こりません。一方、COM STOPの方では、割込みは発生するが、これを受けつけ、実際の処理ルーチンへ行くのを保留しておくのです。ですからCOM STOPを解除した段階で受けつけられ、処理ルーチンへ制御を移します。この時、COM OFFを実行すれば保留されていた割込はキャンセルされます。

COM STOPを解除し、割込み可にするのはCOM ONです。また、割込み保留状態にするには、一旦COM ONにしCOM STOPにしなければなりません。図に示すと次の様になります。

10-4-2 割込みの使用例

前節で、「ボーレイトが速く、かつデータ量が多い時、RS-232Cによる通信では、バッファ・オーバーフローを起こすことがある」と述べました。そして、その対応策として、アクノリッジを返す方法を示しました。この時割込みを使用すると反応が速く、送受双方の処理速度が向上します。

次に示すプログラム（10-4-2）は割込み処理でアクノリッジを返すもので、処理速度向上の為、受信データをキューバッファに入れ、メインルーチンでは、このキューバッファから読み込むようにしています。

(プログラム10-4-2)

```
100 N=10:GOTO 200
110 '***** Interrupt routine *****
120 COM OFF
130 LINE INPUT #1,A$(WSP)                        ' Put to queue
140 WSP=(WSP+1) MOD N
150 IF ((WSP+1) MOD N)=RSP THEN FULL=-1:RETURN ELSE FULL=0
160 PRINT #1,CHR$(4)
170 COM ON
180 RETURN
200 '******* MAIN *******
210 ON COM GOSUB 110
230 OPEN "com:E83XS" AS #1
240 COM ON
250 '
260 IF RSP=WSP MOD N THEN 300
270 A$=A$(RSP):RSP=(RSP+1) MOD N                 ' Get from queue
280 IF FULL THEN GOSUB 150
290 PRINT A$
300 FOR I=0 TO 100: PRINT "..";  :NEXT I         ' Dumy loop
310 PRINT
320 GOTO 260
```

第11章　漢　字

第11章　漢　字

11-1. 漢字ROMボード

PC-8801では、日本語表示のためのオプションとして、漢字ROMボードを装着することができます。

この節では、漢字ROMボードの応用のために、その機能仕様と漢字フォントのフォーマットについて解説します。

11-1-1 ハード仕様

- ROMの構成　　128Kビットマスク ROM×8個（16ビット×64Kワード）
- アクセス方法　　I/Oポートを通してのアクセス

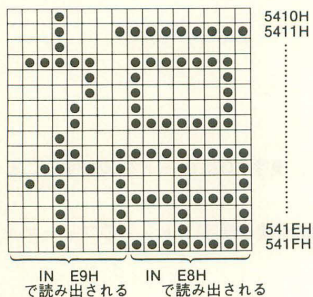
- I/Oポート

E8H	OUT	漢字ROMアドレスの指定(下位8ビット)	
	IN	漢字フォントデータの読み出し(下位8ビット)	
E9H	OUT	漢字ROMアドレスの指定(上位8ビット)	
	IN	漢字フォントデータの読み出し(上位8ビット)	
EAH	OUT	漢字ROMの読み出し開始	データは何でもよい
EBH	OUT	漢字ROMの読み出し終了	

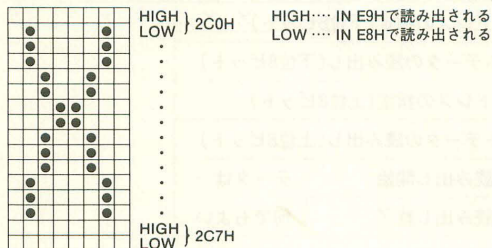
11-1-2 漢字フォントのフォーマット

①漢字（16×16ドット）

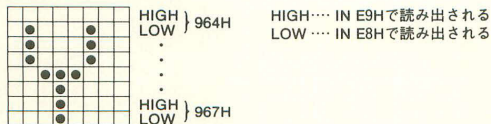
1つの漢字データは、16ワード（1ワードは16ビット）よりなり、次の様な構成になっています。

（例）漢字コード＝4A21H

②半角文字（８×16ドット）

半角文字は、８ワードよりなり、１ワードが、２列分のデータとなっています。

（例）漢字コード＝58H

③1／4角文字　（８×８ドット）

1／4角文字は、４ワードよりなり、半角文字の上半分と思えばよいでしょう。この文字セットは、N_{88}-BASICのキャラジェネ（キャラクタデータがはいっているROM）と同じものです。

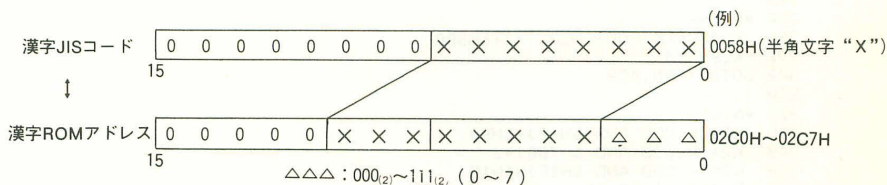
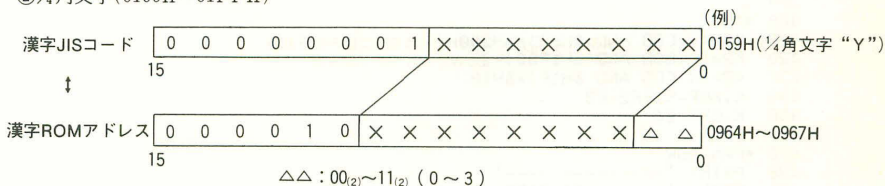
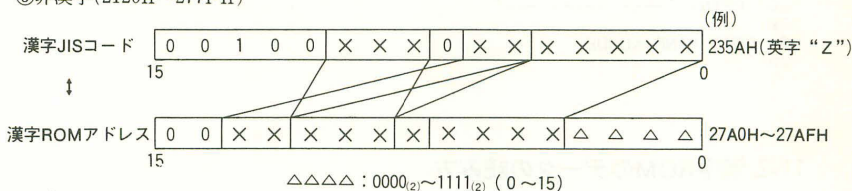
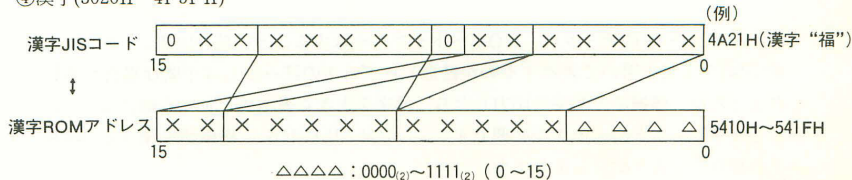
（例）　漢字コード＝159H

11-1-3 漢字ROMのアドレス

漢字ROMアドレスは、16ビットよりなり、これによって漢字ROMボードのアドレス空間64Kワードを指定します。

漢字ROMアドレスと格納されているデータ

〈アドレス〉	〈デ ー タ〉
0000H〜07FFH	半角文字のデータ
0800H〜0BFFH	1/4角文字のデータ
1200H〜3FFFH	非漢字のデータ
4000H〜FFFFH	漢字のデータ

漢字JISコード↔漢字ROMアドレス

①半角文字(0020H〜00FFH)

②¼角文字(0100H〜01FFH)

③非漢字(2120H〜277FH)

④漢字(3020H〜4F5FH)

このように、漢字JISコードから漢字ROMアドレスに変換するのはめんどうな作業です。そこで、BASICで書いた変換プログラムをあげておきます。

```
100 '
110 '    KANJI JIS Code => KANJI ROM Address
120 '
130 *INP.KCODE
140  INPUT "KANJI JIS Code ? &H",KC$
150  K.COD=VAL("&H"+KC$)
160   IF K.COD>=  &H20 AND K.COD<=  &HFF THEN *H.CHR
170   IF K.COD>= &H100 AND K.COD<= &H1FF THEN *F.CHR
180   IF K.COD>=&H2120 AND K.COD<=&H277F THEN *N.KNJ
190   IF K.COD>=&H3020 AND K.COD<=&H4F5F THEN *KNJ
200 GOTO *INP.KCODE
210 '
220 *H.CHR
230  K.ADR=K.COD*8
240  K.BYT=8
250 GOTO *PRN.ADR
260 '
270 *F.CHR
280  K.ADR=(K.COD-&H100)*4+&H800
290  K.BYT=4
300 GOTO *PRN.ADR
310 '
320 *N.KNJ
330  K1=(K.COD AND &H60)*&H80
340  K2=(K.COD AND &H700)*2
350  K3=(K.COD AND &H1F)*&H10
360  K.ADR=K1+K2+K3
370  K.BYT=16
380 GOTO *PRN.ADR
390 '
400 *KNJ
410  K1=(K.COD AND &H60)*&H200
420  K2=(K.COD AND &H1F00)*2
430  K3=(K.COD AND &H1F)*&H10
440  K.ADR=K1+K2+K3
450  K.BYT=16
460 '
470 *PRN.ADR
480  PRINT "--------------"
490  PRINT "KANJI JIS CODE : &H";HEX$(K.COD)
500  PRINT "KANJI ROM ADRS : &H";HEX$(K.ADR)" - &H";HEX$(K.ADR+K.BYT-1)
510  PRINT "--------------"
520 '
530 GOTO *INP.KCODE
540 '
```

11-2. 漢字ROMのデータの読み方

漢字ROMボードのデータは、I/Oポートをアクセスすることで読み出すことができます。

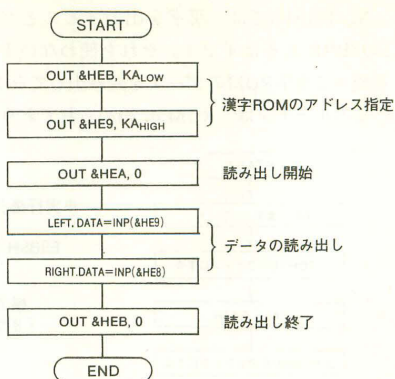
漢字はPUT文を用いて表示するのが普通で、データの読み出しは不要な場合が多いかもしれませんが、機械語で漢字を出力したり、漢字を大きく表示したりする場合などは、どうしても漢字フォントのデータが必要です。ここでは、BASICと、機械語を使っての漢字フォントの読み出し方を紹介しましょう。

11-2-1 BASICを使って

まずは、次のフローチャートに従って、BASICによる漢字フォントデータ読み出しプログラムを作ってみます。

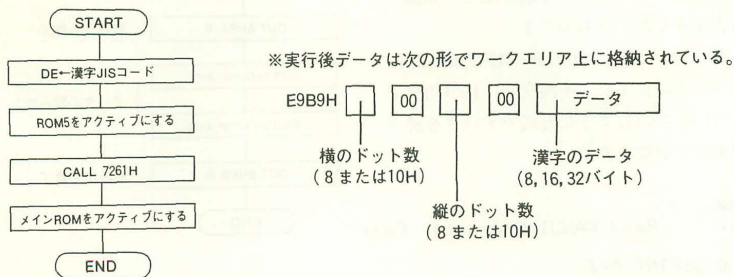
このプログラムは、漢字JISコードを入力すると、そのデータを読み出し、画面上に表示するというものです。

漢字コードは、3021H〜4F53Hと制限がついていますが、半角文字、1/4角文字についても同じような方式でデータを読み出すことができます。

START
OUT &HEB, KA_{LOW}
OUT &HE9, KA_{HIGH} } 漢字ROMのアドレス指定
OUT &HEA, 0 　読み出し開始
LEFT. DATA=INP(&HE9)
RIGHT.DATA=INP(&HE8) } データの読み出し
OUT &HEB, 0 　読み出し終了
END

```
100 '
110 '    Read KANJI Character Font
120 '
130 DEFINT A-Z
140 WIDTH 40,25
150 '
160 INPUT "KANJI Code (&H3021 - &H4F53)";K.CODE
170 IF K.CODE<&H3021 OR K.CODE>&H4F53 THEN 160
180 '
190 GOSUB *KC.TO.KA
200 '
210 FOR KA=K.ADRS TO K.ADRS+15
220   KA.L=PEEK(VARPTR(KA))
230   KA.H=PEEK(VARPTR(KA)+1)
240   OUT &HE8,KA.L
250   OUT &HE9,KA.H
260   OUT &HEA,0
270   L.DAT=INP(&HE9)
280   R.DAT=INP(&HE8)
290   OUT &HEB,0
300   GOSUB *PRINT.PAT
310 NEXT KA
320 '
330 END
340 '
350 *KC.TO.KA    :' Get KANJI-ROM Address from KANJI Code
360  KA1=(K.CODE AND &H60)*&H200
370  KA2=(K.CODE AND &H1F00)*2
380  KA3=(K.CODE AND &H1F)*&H10
390  K.ADRS=KA1+KA2+KA3
400 RETURN
410 '
420 *PRINT.PAT
430  PRINT HEX$(KA)"H : ";
440  DAT=L.DAT : GOSUB *PRINT.DOT
450  DAT=R.DAT : GOSUB *PRINT.DOT
460  PRINT
470 RETURN
480 *PRINT.DOT
490  FOR B=7 TO 0 STEP -1
500    IF DAT AND (2^B) THEN PRINT "■"; ELSE PRINT " ";
510  NEXT B
520 RETURN
```

11-2-2 N88-BASIC ROMルーチンを使って

N88-BASICでは、漢字を出力することができるわけですから、そのためのルーチンがROM内にあるはずです。それを使わない手はないかというので、ここではROMルーチンを使って漢字ROMのデータを読み出してみましょう。

このルーチンは、ROM5に納められており次のように使います。

実は、このルーチンは、PUT KANJIの一部で、余分な処理まで行なってしまいますが、実用上は問題ありません。なお、このワークエリア(E9B9H〜)は、行入力バッファも兼ねていますので、ROMルーチンから戻った後、INPUT文などを実行すると、読み出したデータは消されてしまいますから、読み出した後は、別のところへ移しておく方が安全です。

次のプログラムは、漢字コードを入力すると、そのデータを16進で出力するというもので、半角文字、1/4角文字でも使えます。

```
100 '
110 '   Read KANJI Character Font
120 '                  ( Using N88-BASIC ROM Routine )
130 '
140 DEFINT A-Z
150 '
160 DEF USR=&HF2E0
170 FOR I=&HF2E0 TO &HF2F2
180   READ D$ : POKE I,VAL("&H"+D$)
190 NEXT
200 DATA 7E,23,66,6f,EB,F3,3E,FE,D3,71,CD,61,72,3E,FF,D3
210 DATA 71,FB,C9
220 '
230 DT.TOP=&HE9B9
240 '
250 INPUT "KANJI JIS Code ? &H",KC$
260 KC=VAL("&H"+KC$)
270 '
280 DUM=USR(KC)
290 '
300 PX=PEEK(DT.TOP) : PY=PEEK(DT.TOP+2)
310 DT.TOP=DT.TOP+3
320 FOR I=1 TO (PX/8)*PY
330  PRINT RIGHT$("0"+HEX$(PEEK(DT.TOP+I)),2)" ";
340 NEXT I
```

11-3. ROLL文

漢字の出力はグラフィック画面に対して行われますので、テキスト画面のように自動的にスクロールするようなことはありません。そこで必要となってくるのがROLL文です。

ROLL文は、たいへん便利なコマンドですが、残念ながらN_{88}-DISK-BASICでしか使うことができません。また、処理速度の遅さも気になります。

そこで、この節では、N_{88}-BASICでの高速ROLL機能を付加するプログラムを紹介します。

```
100 '
110 '    SCROLL COMMAND for 640 x 400 dot mode
120 '
130 '    DUMMY = USR(SCROLL.BYT)
140 '
150  DEFINT A-Z
160  RESTORE *ML.DATA
170  SADRS=VARPTR(#0)+9
180  ADRS=SADRS
190 '
200 *WRITE.DATA
210  READ DATUM$
220  IF DATUM$="END" THEN *FINISH
230  POKE ADRS,VAL("&H"+DATUM$)
240  ADRS=ADRS+1
250 GOTO *WRITE.DATA
260 '
270 *FINISH
280  DEF USR=SADRS
290  PRINT "Complete."
300  END
310 '
320 *ML.DATA
330  DATA 7E,23,66,6F,E5,E5,11,00,C0,19,E3,E5,21,80,3E,C1
340  DATA B7,ED,42,4D,44,E1,E5,D5,C5,F3,D3,5C,ED,B0,D3,5F
350  DATA FB,E1,C1,C5,E5,11,80,FE,7C,F6,C0,67,F3,D3,5D,0A
360  DATA D3,5C,77,D3,5F,FB,23,03,E7,20,F1,C1,D1,E1,C5,F3
370  DATA D3,5D,ED,B0,D3,5F,FB,E1,7C,F6,C0,67,5D,54,1B,C1
380  DATA F3,D3,5D,36,00,ED,B0,D3,5F,FB,C9
390  DATA END
```

上のプログラムを実行すると、あとは

　DUMMY＝USR(SCROLL.BYT)

で、グラフィック画面をスクロールさせることができます。

SCROLL.BYTは、整数型でなければならず

　ROLL 18 ↔ DUMMY＝USR(18＊80)

に対応し、引数を80の倍数以外にすると、斜めにスクロールさせたりすることも可能です。

なお、機械語プログラムは、リロケータブルになっていますので、C000H番地より前であれば、メモリ上のどこにでも置けます。この例では、＃０のI/Oバッファを使ってみました。

最後に、このROLL機能を使った例をあげておきます。

```
100 '
110 '   SCROLL DEMO
120 '
130  DEFINT A-Z
140  CONSOLE ,,0
150  SCREEN 2,2 : CLS 3 : SCREEN ,0
160 '
200  FOR I=&H3000 TO &H4F00 STEP &H100
210   FOR J=&H21 TO &H7E
220    K.CODE=I+J
230    PUT(X,380),KANJI(K.CODE),PSET
240    X=X+20
250    IF X>620 THEN X=0 : GOSUB *SCROLL
260   NEXT J
270  NEXT I
275 END
280 '
300 *SCROLL
310  SCREEN ,2 : DUMMY=USR(1600) : SCREEN ,0
320 RETURN
```

第12章　ランダム・テクニック

第12章　ランダム・テクニック

12-1. DMAをストップさせて実行速度アップ

DMAとはDirect Memory Accessの略で、CPUを介さずにメモリーをアクセスすることです。N88-BASICでは画面の表示と8インチディスクのアクセスに使われています。このうち実行速度を低下させているのはテキスト画面の表示だけです。グラフィック画面の表示は通常、CPUの用いるバスとは切りはなされた状態で行われていますのでCPUの処理速度には影響ありませんし、ディスクのDMAはディスクをアクセスする時だけですのであまり関係ありません。

さて、テキスト画面表示のDMAを止めるのは簡単で、OUT 104,0を実行するだけです。テキスト画面が表示されなくなりますが、実行速度は30％くらい速くなります。グラフィック画面の表示はそのままです。ところがDMAを戻すのがやっかいです。WIDTHコマンドを実行するとテキスト画面は表示されるようになりますが､画面がクリアされてしまいます。画面を表示させるためにはDMAコントローラと、CRTコントローラの設定をしなくてはなりません。幸いROM内にそのルーチンがありますのでそれを利用することにします。

モニタで次のように打ち込んで下さい。プログラムはリロケータブルですのでF260H番地からでなくてもかまいません。

```
F260 21 5B 70 3A 88 EF FE 15 38 03 21 66 70 3A 89 EF
F270 FE 50 F3 CD D1 6F FB C9
```

実行するにはUSRまたはCALL文で呼びます。引数はありません。

12-2. 配列データ高速読み込み

数値型の配列のデータを読み込むには、通常次の方法があります。

①プログラム中にデータ文として存在するデータをREAD文で読む。

②ファイルとして存在するデータをINPUT♯n文で読む。

どちらの方法をとったとしてもデータが大量な場合にはけっこう時間がかかります。このような場合には、データを機械語ファイルとしてしまっておいて、BLOAD文で一気に読み込むことができます。

まず配列データをBSAVEします。開始番地はVARPTR（〈配例名〉(0)）です。１次元配列の時OPTION BASEの値はEC1FH番地に入っていますから、VARPTR（〈配列名〉(PEEK(&HEC1F))）としてもけっこうです。データの長さは（要素の数）×（１要素のバイト数）です。

（要素の数）は、

「添字の最大値＋１－OPTION BASE」をすべてかけ合わせたものです。

（１要素のバイト数）は、

整数型…２バイト
単精度…４バイト
倍精度…８バイト

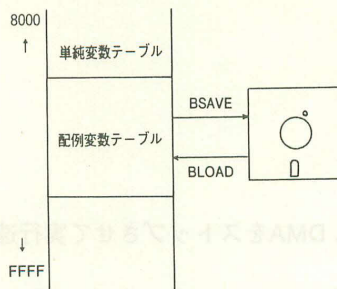
です。たとえば、DIM A%(５，６，７)，OPTION BASE０の時は、

BSAVE"〈ファイルディスクリプタ〉"，VARPTR(A%(０，０，０))，６*７*８*２

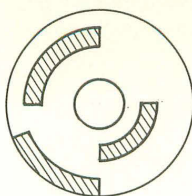
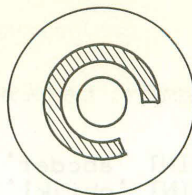
となります。

データをBLOADする時も同じです。まずDIMでBSAVEした時と同じ大きさの配列を宣言します。次にBLOADします。先ほどの例では、

BLOAD"〈ファイルディスクリプタ〉",VARPTR(A%(０，０，０))

となります。

12-3. xfilesでクランチを

システムディスクに入っている「xfiles.n88」は異なるタイプのディスク間でのバックアップなどに使われますが、同じタイプのディスクであっても利用法があります。

多数のファイルの生成、削除を繰り返していると、ファイルデータがディスク上のあちこちに分散してしまいます。このようなディスクをxfilesで他のディスクに移すと、１ヵ所にかたまって記録されるようになります。こうすると、ファイルへのアクセス時にヘッドがあまり動く必要がなく、その分だけアクセスが早くなります。

12-4. 行番号 0

BASICプログラムの行番号は、1～65529までの整数というのが普通ですが、N88-BASICでも、行番号0が使えます。ただし、スクリーンエディット時には、行番号0は使えませんので、次のように、間接的に行番号0のテキストを作ります。

まず、1以上の行番号を持ったテキストを入力します。次に、RENUM 0を実行すれば、行番号0のテキストができるわけです。

```
10 '*********************
renum 0
Ok
list
0 '*********************
Ok
```

こうやって作られた行番号0のテキストはスクリーエディタで修正することも削除することもできません(DELETE 0は使えます)。

この場合、1つの行しか行番号を0にできませんが、次の方法では、複数行の行番号を0にすることが可能です。

これは、RAM上にあるプログラムテキストの行番号を、直接0にしてしまうというもので、一つ間違うとプログラム自体をこわしてしまう可能性がありますので、注意が必要です。

具体的な例で見てみましょう。

```
10 PRINT "abcdef"
20 PRINT "ghijkl"
30 PRINT "mnopqr"
```

モニタモードで、メモリ内容をダンプし、行番号の部分を0にします。

```
h]o70,0
h]d8000,802f
8000 00 10 00 0A 00 91 20 22 61 62 63 64 65 66 22 00
8010 1F 00 14 00 91 20 22 67 68 69 6A 6B 6C 22 00 2E
8020 00 1E 00 91 20 22 6D 6E 6F 70 71 72 22 00 00 00

mon

h]s8003
8003 0A-00
h]s8012
8012 14-00
h]s8021
8021 1E-00
h]^b
Ok
```

これで完成です。もちろん実行することも可能です。

```
list
0 PRINT "abcdef"
0 PRINT "ghijkl"
0 PRINT "mnopqr"
Ok
run
abcdef
ghijkl
mnopqr
Ok
```

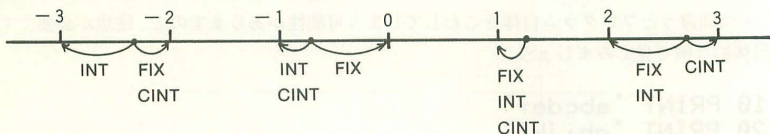
12-5. FIX, INT, CINT

この3つはどれも実数型を整数に変換する関数ですが、微妙な違いがあります。

●FIX……整数部分をとる関数で1.5→1, −2.3→−2となります。

●INT……その値よりも小さい整数の中で最大のものをとる関数です。1.5→1となりますが、−2.3→−3となります。

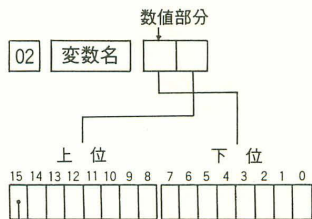
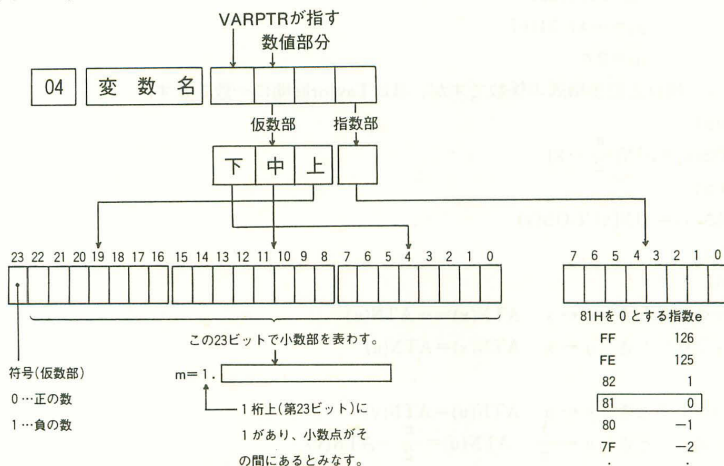
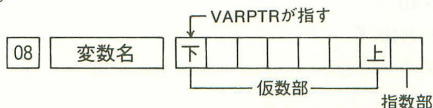
●CINT…小数部分を四捨五入します。1.5→2, −2.3→−2となります。

これらを図に示すとこのようになります。

12-6. 数値と文字列の変換

数値を文字列に変換する関数にはSTR$をはじめとしてHEX$, OCT$, CHR$, MKI$, MKS$, MKD$とたくさんありますが、それらの一覧表を上げます。

12-7. 数値の内部表現

CALL文では変数のアドレスが渡されますが、数値の内部表現がわからなければ利用のしようがありません。またVARPTRを用いて変数の値を操作する時も同様です。そこで数値の内部表現を下に示します。

〔整数型〕

〔単精度型〕

〔倍精度型〕

仮数部が7バイト（56ビット）に増えただけで単精度と同じです

12-8. 三角関数の求値法

N_{88}-BASICインタプリタがどのようにして数値関数の値を求めているのかというと、すべて近似式によって計算しているのです。ここではN_{88}-BASIC,N-BASICが使用している三角関数（単精度）を求めるアルゴリズムを紹介します。

①SIN(x)

$w \leftarrow x/2\pi$

$u \leftarrow w - INT(w)$

もし、

$0 \leqq u < 0.25$ のとき $v \leftarrow u$

$0.25 \leqq u < 0.75$ のとき $v \leftarrow 0.5 - u$

$0.75 \leqq u < 1$ のとき $v \leftarrow u - 1$

$s \leftarrow v^2$

$SIN(x) = v \times (a_1 s^4 + a_2 s^3 + a_3 s^2 + a_4 s + a_5)$

ただし、$a_1 = 39.71067$

$a_2 = -76.57498$

$a_3 = 81.60223$

$a_4 = -41.34167$

$a_5 = 2\pi$

これらは、最良近似多項式の係数ですが、ほぼTaylor展開に一致します。

②COS(x)

$COS(x) = SIN(\frac{\pi}{2} - \bar{x})$

③TAN(x)

$TAN(x) = SIN(x)/COS(x)$

④ATN(x)

もし、

$x < 0$ のとき $u \leftarrow x$, $ATN(x) = -ATN(u)$

$x \geqq 0$ のとき $u \leftarrow x$, $ATN(x) = ATN(u)$

もし、

$u < 1$ のとき $v \leftarrow u$, $ATN(u) = ATN(v)$

$u \geqq 1$ のとき $v \leftarrow \frac{1}{u}$, $ATN(u) = \frac{\pi}{2} - ATN(v)$

$s \leftarrow v^2$

$ATN(v) = v \times (a_1 s^8 + a_2 s^7 + a_3 s^6 + a_4 s^5 + a_5 s^4 + a_6 s^3 + a_7 s^2 + a_8 s + a_9)$

ただし、$a_1 = 2.866226 \times 10^{-3}$

$a_2 = -1.616574 \times 10^{-2}$

$a_3 = 4.290961 \times 10^{-2}$

$a_4 = -7.528964 \times 10^{-2}$

$a_5 = 0.1065626$

$a_6 = -0.1420890$

$a_7 = 0.1999355$

$a_8 = -0.3333315$

$a_9 = 1$

12-9. CTRL+J

```
10 DEFINT A-Z
  :SCREEN ,2
  :CLS 3
  :SCREEN 0,0
  :WHILE 1
  : X=RND*560
  : Y=RND*160+20
  : L=RND*60+40
  : LINE(X,Y)-(X+L,0),INT(RND*7+1),BF
  : LINE(X,Y)-(X+L,0),0,B
  :WEND
```

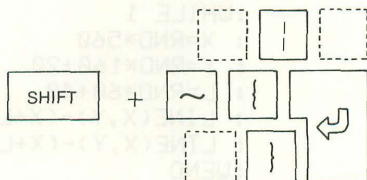
このプログラム、どうやって入力したと思いますか。全部で11行、WIDTH40で入力したとして400文字以上になります!?実はこれ、 CTRL ＋Jを使ったんです。改行する時にスペースを入れていって次の行までもっていくのではなくて、 CTRL ＋Jを押します。するとカーソルが次の行の先頭に移動し、一見⏎キーか STOP キーを押したようになりますが、そうではなくて、前の行の続きを入力できます。ためしに DEL キーを押してみると前の行の最後にカーソルが移動し、前の行から続いていることがわかります。

また CTRL ＋Jは2つの行をつなげる時にも使います。たとえば10行と20行をつなげたい場合にはリストをとった後、10行の最下行にカーソルを持ってきて CTRL ＋Jを押します。カーソルは「20」のところに移りますから行番号を消してかわりに「：」を打ちます。必要があればDELキーで前の行にもっていった後、⏎キーを押します。20行はまだそのまま残っていますから20⏎とやって20行を消します。

```
ここにカーソルを          10························
持ってきてCTRL+J  →  ■························
を押す                    20························
```

以上のような CTRL ＋Jの使い方はノーマルモードでの話で、インサートモードでは CTRL ＋Jは行の分割になります。

12-10. キートップにない文字の入力

キートップに刻まれていない文字の中で、キーボードから入力できる文字はいろいろとあります。アルファベットの小文字、グラフィック文字などは当然のことなのですが、意外と知られていないものに次のキーがあります。これらの文字は、次のキーに対応し、キーボードから直接入力することができるものです。

12-11. 文字が曲がる!?

```
10 CONSOLE ,,,0
20 WIDTH 80,25
30 FOR I=0 TO 22
40   PRINT STRING$(80,"X");
50 NEXT
60 OUT &H30,2:'
70 OUT &H30,3:'
80 GOTO   60
```

何も考えずに上のプログラムをそのまま入力してRUNして下さい。高解度ディスプレイを御使用の方は80行のGOTOと60との間を4文字ほど広げて下さい。また各PC-8801の水晶の発振周波数の違いにより、GOTOと60との間のスペースの数に調整が必要なことがあります。

なおブレークした時に画面がおかしな状態になったらWIDTHコマンドを実行して下さい。また表示させる文字を色々変えてみると面白いですよ。

12-12. N-BASICモードでcas 1 （1200ボー）を使う。

N_{88}-BASICモードでは、カセットファイルに600ボー(cas2)と1200ボー(cas1)の２種の転送速度が使えましたが、N-BASICモードでは、600ボーのみしか使用できず、長いデータを作った時などカセット入出力に時間がかかってしまいます。そこで次のプログラムを実行させると、N-BASICモードで600ボーと1200ボーの両方が使えます。転送速度の切り換えはCMDコマンドを使用します。

CMDB ‥‥ 600ボー

CMDA ‥‥ 1200ボー

このプログラムは、N_{88}-BASICモードのcas1で作ったデータファイルを読む時にも使用できます。なお、プログラムファイルを1200ボーでCSAVE、CLOADすることもできますが、CLOADの時、このプログラムを実行していなければ全く読めません。

```
10 '████ 1200 bps for N-Basic mode ████
20 CLEAR 300,&HE9C4
30 FOR I=0 TO 58
40 READ A$
50 POKE &HE9C5+I,VAL("&H"+A$)
60 NEXT I
70 POKE &HF0FC,&HC3
80 POKE &HF0FD,&HC5
90 POKE &HF0FE,&HE9
100 DATA 7E,D6,42,20,0A,3E,C9,32,B6,F1,32
110 DATA B9,F1,D7,C9,3C,C2,DF,3B,E5,21,EA,E9,22,B7,F1,21
120 DATA F5,E9,22,BA,F1,3E,C3,E1,18,E2,F1,3A,66,EA,E6,0F
130 DATA F6,18,C3,FD,0B,F1,3A,66,EA,E6,0F,F6,1C,C3,50,0C
```

12-13. ソフトファンクションキー

ソフトファンクションキーとはその名の通りソフトウェアでファンクションキーの役割をさせようというものです。その方法は次の通りです。

1．メモリ中に文字列を用意します。文字列の最後は00Hで終っていなければなりません。
2．E6CDH番地に00Hでない値を書き込みます。
3．F003,4H番地に文字列の先頭アドレスを下位、上位の順に入れます。
4．E6CEH番地に00Hでない値を書き込みます。
5．F00CH番地にファンクションキー割込みを定義していないファンクションキー番号−1を書き込みます。
6．E6CDH番地に00Hを書き込みます。

例を上げてみましょう。

```
list
10 A$="This is a pen."
20 P=VARPTR(A$) : KEY OFF
30 POKE &HE6CD,255
40 POKE &HF003,PEEK(P+1) : POKE &HF004,PEEK(P+2)
50 POKE &HE6CE,255
60 POKE &HF00C,0
70 POKE &HE6CD,0
Ok

run
Ok
This is a pen.
```

このうち、肝心なのは40行と50行です。30行はキー入力を止めるためで、70行は再開するためです。そうしないと実行中にキーを押すとうまく動かないことがあるからです。60行は、割込みを起さないためです。ここに、割込みが定義されていてON状態であるキー番号－1をPOKEすると、そのキーの割込みが起ってしまい、ファンクションキーの働きをしません。しかしKEY OFFがしてあれば60行はなくてもかまいません。

この方法を使うと、プログラムをPRINT文で書いた後、カーソルをその行にもって行き⏎を押したことにすれば自動的にプログラムを増やすことができます。DATA文の自動作成などに有効です。

12-14. 機械語割込み

N_{88}-BASICではZ-80はインタラプトモード2に設定されています。割込みレベルは8レベルで、割込みテーブルはF300H番地から次のように入っています。

N_{88}-BASICの割り込みテーブル表

優先順位	アドレス	アドレステーブルの内容	チャネル	用　途	BASICでの使用
高	F300 1	EA E7	RXRDY	CMT RS-232C 受信割り込み	YES
↑	F302 3	08 E8	VRTC	画面終了割り込み	YES
	F304 5	0F E8	CLOCK	リアルタイムクロック(1/600S)	YES
	F306 7	14 E8	INT3(4)	ユーザ割り込み	NON
	F308 9	14 E8	INT4(3)	〃	NON
	F30A B	14 E8	INT5(2)	〃	NON
↓	F30C D	1A E8	FDINT1	FDD用リザーブ	YES
低	F30E F	20 E8	FDINT2	FDD用リザーブ	YES

このうちユーザーが使用できるのはINT3～5の3チャネルです。他の割込みも使えないことはありませんが、N_{88}-BASICで使用していますので、うまくやらないとN_{88}-BASICが動かなくなる恐れがあります。N-BASICでは割込みを使用していませんのでN-BASICモードではすべての割込みを使用することができます。

N_{88}-BASICでは割込みルーチンのエントリはRAM上に置かれています。ここには、割込みレベルを割込みコントローラにセットし、メモリバンクをN_{88}-ROMにもどして、本当の割込み処理ルーチンをCALLするプログラムがすでに書き込まれています。これはRAM上にあるので書きかえることも可能ですが、やはりBASICインタプリタが使用しているので十分な注意が必要です。

N_{88}-BASICでの割込み処理ルーチンは以下のとおりです。

●RXRDY

テーブルアドレス: F300H

割込みルーチンエントリ: E7EAH

ROM内割込み処理ルーチン: 3167H

シリアルインターフェースから受取ったデータを入力バッファに入れます。RS-232Cの場合はXパラメータの処理も行います。

●VRTC

テーブルアドレス: F302H

割込みルーチンエントリ: E808H

ROM内割込み処理ルーチン: 3080H

カーソルのON/OFF、ライトペン入力処理、キー入力処理を行います。

●CLOCK

テーブルアドレス: F304H

割込みルーチンエントリ: E80EH

ROM内割込み処理ルーチン: 4143H

INPUT WAITの時間処理、ON TIME$ GOSUBの時間処理、FDDの時間処理を行います。

●FDINT1

テーブルアドレス: F30CH

割込みルーチンエントリ: E81AH

RO内割込み処理ルーチン: 3CB9H

5インチDMAタイプのFDDのステータス読み込みを行います。

●FDINT2

テーブルアドレス: F30FH

割込みルーチンエントリ: E820H

RO割込みルーチン: 3CACH

8インチDMAタイプのFDDのステータス読み込みを行います。

付　　録

付録1　機械語サブルーチン・ソースリスト

ここでは、各章で使われている機械語、プログラムのソースリストを掲載します。具体的な使い方、その他は、各章を参照してください。

1. オールマイティ PEEK, POKE

```
              ;------------------------
              ; PC-TechKnow 8800 VOL.1
              ;  <<  PEEK POKE  >>
              ;  (C) 1982 by Radix
              ;------------------------
*
*     PEEK ( <ADDRESS> [,<BANKNAME>])
*     POKE <ADDRESS>,<DATA> [,<BANKNAME>]
*
*                        00  60    80      84  C0                 MODE
*       <BANKNANE> : NONE  N88,N88 ,(WINDOW),RAM,RAM                5
*                    " "     SAME                                   5
*                    "T"   RAM,RAM ,  RAM    ,RAM,RAM               3
*                    "N"   N  ,N   ,  RAM    ,RAM,RAM               4
*                    "n"   N88,ROM5,(WINDOW),RAM,GRAMn (n=0 to 2)  0,1,2
*
*     [ INIT ]
*       G E400
*
              ;
                       ORG   0E400H
              ;
0B06          FCERR: EQU  0B06H           ;ILLEGAL FUNCTION CALL
11D3          FRMEVL:EQU  11D3H           ;EVALUATE EXPR.
18A3          GETBYT:EQU  18A3H           ;GET PARAMETER(0-255) to Acc
1B93          GETADR:EQU  1B93H           ;GET ADRS to DE (-23768 to 65535)
56C9          CSFRFA:EQU  56C9H           ;CHECK STRING & FREFAC
              ;
8450          SWAREA:EQU  8450H
              ;
E6C2          SVCRTC:EQU  0E6C2H          ;IMAGE OF PORT31H
EABD          VALTYP:EQU  0EABDH          ;TYPE of FAcc
EC41          FACLO: EQU  0EC41H          ;FAC ADRS
              ;
              ;---- INITIALIZE ----
              ;
E400          INIT:
E400 3A74E5            LD    A,(WED27)
E403 B7                OR    A
E404 2034              JR    NZ,INIT9
              ;
E406 F3                DI
E407 2127ED            LD    HL,0ED27H
E40A 1174E5            LD    DE,WED27
E40D 010300            LD    BC,3
E410 EDB0              LDIR
              ;
E412 219FED            LD    HL,0ED9FH
E415 1177E5            LD    DE,WED9F
E418 010300            LD    BC,3
E41B EDB0              LDIR
              ;
E41D 3EC3              LD    A,0C3H
E41F 3227ED            LD    (0ED27H),A
E422 329FED            LD    (0ED9FH),A
E425 3A27ED            LD    A,(0ED27H)
E428 213CE4            LD    HL,PEEKQ
E42B 2228ED            LD    (0ED28H),HL
E42E 21EBE4            LD    HL,POKE
E431 22A0ED            LD    (0EDA0H),HL
E434 3E6C              LD    A,6CH
E436 3227E8            LD    (0E827H),A
E439 FB                EI
E43A          INIT9:
E43A FF                RST   38H
```

```
E43B C9              RET
                ;
                ;------- PEEK -------
                ;
E43C            PEEKQ:                            ;from ED27
E43C 3C              INC  A
E43D 2804            JR   Z,PEEKQ2
E43F 3D              DEC  A
E440 C374E5          JP   WED27
E443 23         PEEKQ2:INC HL
E444 7E              LD   A,(HL)
E445 FE97            CP   97H
E447 2804            JR   Z,PEEK
E449 2B              DEC  HL
E44A 3EFF            LD   A,255
E44C C9              RET
                ;
E44D            PEEK:
E44D F1              POP  AF
E44E CD77E5          CALL WED9F
E451 3E05            LD   A,5
E453 F5              PUSH AF              ;SAVE MODE
E454 D7              RST  10H
E455 CF              RST  8H
E456 28              DB   '('
E457 CD931B          CALL GETADR
E45A D5              PUSH DE              ;SAVE ADRS
E45B 2B              DEC  HL
E45C D7              RST  10H
E45D FE29            CP   ')'
E45F 2809            JR   Z,PEEK10        ;GETEND
E461 CF              RST  8H
E462 2C              DB   ','
                ;       --- BANKNAME ---
E463 CDBCE4          CALL GETBN
E466 D1              POP  DE              ;ADRS
E467 C1              POP  BC              ;POP OLD MODE
E468 F5              PUSH AF              ;SAVE NEW MODE
E469 D5              PUSH DE              ;ADRS
                ;
E46A            PEEK10:
E46A CF              RST  8H
E46B 29              DB   ')'
E46C D1              POP  DE              ;ADRS
E46D ED5371E5        LD   (ADRS),DE
E471 F1              POP  AF              ;MODE
E472 3273E5          LD   (MODE),A
E475 E5              PUSH HL              ;TEXT POINTER
E476 3E02            LD   A,2
E478 32BDEA          LD   (VALTYP),A      ;INTEGER
E47B 2A71E5          LD   HL,(ADRS)
E47E CD28E5          CALL IFSWAP
E481 CD99E4          CALL SETMDE
E484 4E              LD   C,(HL)
E485 D35F            OUT  (5FH),A
E487 3EFF            LD   A,255
E489 D371            OUT  (71H),A
E48B 3AC2E6          LD   A,(SVCRTC)
E48E D331            OUT  (31H),A
E490 FB              EI
E491 69              LD   L,C
E492 2600            LD   H,0
E494 2241EC          LD   (FACLO),HL
E497 E1              POP  HL
E498 C9              RET
                ;
                ;  ---- SET MODE ----
                ;
E499            SETMDE:
```

```
E499 F3                DI
E49A 3A73E5            LD    A,(MODE)
E49D D603              SUB   3
E49F 3005              JR    NC,SMDE1
E4A1 3EFE              LD    A,0FEH
E4A3 D371              OUT   (71H),A
E4A5 C9                RET
E4A6 2008     SMDE1:   JR    NZ,SMDE2
E4A8 3AC2E6            LD    A,(SVCRTC)          ;TEXT
E4AB F602              OR    2
E4AD D331              OUT   (31H),A
E4AF C9                RET
E4B0 3D       SMDE2:   DEC   A
E4B1 C0                RET   NZ
E4B2 3AC2E6            LD    A,(SVCRTC)          ;N-BAS
E4B5 F604              OR    4
E4B7 E6FD              AND   0FDH
E4B9 D331              OUT   (31H),A
E4BB C9       SMDE3:   RET
              ;
              ;   ---- GET BANKMANE ----
              ;
E4BC          GETBN:
E4BC CDD311            CALL  FRMEVL
E4BF E5                PUSH  HL                      ;TEXT POINTER
E4C0 CDC956            CALL  CSFRFA
E4C3 7E                LD    A,(HL)
E4C4 B7                OR    A
E4C5 CA060B            JP    Z,FCERR
E4C8 23                INC   HL
E4C9 5E                LD    E,(HL)
E4CA 23                INC   HL
E4CB 56                LD    D,(HL)
E4CC 1A                LD    A,(DE)
E4CD 0E05              LD    C,5
E4CF FE20              CP    ' '
E4D1 2815              JR    Z,GETBN1
E4D3 0D                DEC   C
E4D4 FE4E              CP    'N'
E4D6 2810              JR    Z,GETBN1
E4D8 0D                DEC   C
E4D9 FE54              CP    'T'
E4DB 280B              JR    Z,GETBN1
E4DD D630              SUB   '0'
E4DF DA060B            JP    C,FCERR
E4E2 FE03              CP    3
E4E4 D2060B            JP    NC,FCERR
E4E7 4F                LD    C,A
E4E8 79       GETBN1:LD      A,C
E4E9 E1                POP   HL
E4EA C9                RET                           ;ACC , C are MODE
              ;
              ;
              ;------- POKE -------
              ;from ED9F ,ADDRESS IS PUSHED
              ;
E4EB          POKE:
E4EB F1                POP   AF
E4EC CD77E5            CALL  WED9F
E4EF CF                RST   8H
E4F0 2C                DB    ','
E4F1 3E05              LD    A,5
E4F3 F5                PUSH  AF                      ;SAVE MODE
E4F4 CDA318            CALL  GETBYT
E4F7 F5                PUSH  AF                      ;DATA
E4F8 2B                DEC   HL
E4F9 D7                RST   10H
E4FA 2809              JR    Z,POKE3
              ;     --- BANK NAME ---
```

```
E4FC CF              RST  8
E4FD 2C              DB   ','
E4FE CDBCE4          CALL GETBN
E501 C1              POP  BC               ;DATA
E502 D1              POP  DE               ;POP OLD MODE
E503 F5              PUSH AF               ;SAVE NEW MODE
E504 C5              PUSH BC               ;DATA
                ;
E505            POKE3:
E505 C1              POP  BC               ;B=DATA
E506 F1              POP  AF
E507 3273E5          LD   (MODE),A
E50A D1              POP  DE
E50B ED5371E5        LD   (ADRS),DE
E50F E5              PUSH HL               ;TEXT POINTER
E510 2A71E5          LD   HL,(ADRS)
E513 CD28E5          CALL IFSWAP
E516 CD99E4          CALL SETMDE
E519 70              LD   (HL),B
E51A D35F            OUT  (5FH),A
E51C 3EFF            LD   A,255
E51E D371            OUT  (71H),A
E520 3AC2E6          LD   A,(SVCRTC)
E523 D331            OUT  (31H),A
E525 FB              EI
E526 E1              POP  HL
E527 C9              RET
                ;
                ; ---- SWAP? ----
                ;
E528            IFSWAP:
E528 3A73E5          LD   A,(MODE)
E52B FE03            CP   3
E52D D0              RET  NC               ;MODE <> G-RAM
E52E 1100C0          LD   DE,0C000H
E531 E7              RST  20H              ;CP HL,DE
E532 D8              RET  C                ;HL < C000H
                ;
E533 F3              DI
E534 78              LD   A,B              ;DATA
E535 215084          LD   HL,SWAREA
E538 117AE5          LD   DE,WKBACK
E53B 010600          LD   BC,6
E53E EDB0            LDIR
                ;
E540 E1              POP  HL               ;RET ADRS
E541 F5              PUSH AF               ;DATA
E542 3ED3            LD   A,0D3H           ;OUT
E544 115084          LD   DE,SWAREA
E547 12              LD   (DE),A
E548 3A73E5          LD   A,(MODE)
E54B C65C            ADD  A,5CH
E54D 13              INC  DE
E54E 12              LD   (DE),A
E54F 23              INC  HL
E550 23              INC  HL
E551 23              INC  HL
E552 13              INC  DE
E553 010300          LD   BC,3
E556 EDB0            LDIR
E558 3EC9            LD   A,0C9H
E55A 12              LD   (DE),A
E55B C1              POP  BC
E55C E5              PUSH HL               ;NEW RETURN ADRS
E55D 2A71E5          LD   HL,(ADRS)
                ;
E560 CD5084          CALL SWAREA
                ;
```

```
E563 C5              PUSH BC
E564 217AE5          LD   HL,WKBACK
E567 115084          LD   DE,SWAREA
E56A 010600          LD   BC,6
E56D EDB0            LDIR
E56F C1              POP  BC
                ;
E570 C9              RET
                ;
                ;------- WORK AREA -------
                ;
E571 0000       ADRS:  DW   0
E573 00         MODE:  DB   0
E574 000000     WED27: DB   0,0,0
E577 000000     WED9F: DB   0,0,0
E57A 00000000   WKBACK:DW   0,0,0
E57E 0000
                ;
E580                 END
```

2.　N_{88}-BASIC復活

```
                ;----------------------
                ; PC-TechKnow 8800 VOL.1
                ; << RECOVER PROGRAM >>
                ;  (C) 1982 by Radix
                ;----------------------

                        ORG  0F260H
                ;
05BD            LINKER:EQU  05BDH
1BBB            ADRLIN:EQU  1BBBH
44D5            MAPTRU:EQU  44D5H
                ;

E658            TXTTAB:EQU  0E658H
EB18            TXTEND:EQU  0EB18H
EB1A            LBLFLG:EQU  0EB1AH
                ;
F260            RECOVR:
F260 2A58E6             LD   HL,(TXTTAB)
F263 3601               LD   (HL),1
                ;
F265 CDBD05             CALL LINKER
F268 CDBB1B             CALL ADRLIN
F26B CDD544             CALL MAPTRU
F26E 23                 INC  HL
F26F 2218EB             LD   (TXTEND),HL
                ;
F272 AF                 XOR  A
F273 321AEB             LD   (LBLFLG),A
                ;
F276 FF                 DB   0FFH
                ;
F277                    END
```

3. アトリビュート・セットサブルーチン

```
                    ;-------------------------
                    ; PC-TechKnow 8800 VOL.1
                    ;  << ATTRIBUTE SET >>
                    ;  (C) 1982 by Radix
                    ;-------------------------
                    ;
                    ;  DUMMY = USR(VRAM.ADRS)
                    ;
                    ;  [[[ RELOCATABLE ]]]
                    ;
                            ORG  0F320H
                    ;
4351                PUTATR:EQU  4351H
                    ;
F320                ATRSET:
F320 0E00                   LD   C,0
F322 7E                     LD   A,(HL)
F323 23                     INC  HL
F324 66                     LD   H,(HL)
F325 6F                     LD   L,A
F326 CD5143                 CALL PUTATR
F329 C9                     RET
                    ;
F32A                        END
```

4.　GET@、PUT@ サブルーチン

```
                    ;------------------------
                    ; PC-TechKnow 8800 VOL.1
                    ;  << GET@,PUT@ SUB >>
                    ;  (C) 1982 by Radix
                    ;------------------------
                    ;
                    ;  [[[ RELOCATABLE ]]]
                    ;
                            ORG   0F2E0H
                    ;
429D                VADDR: EQU   429DH
4350                VRMPUT:EQU   4350H
4452                VRMGET:EQU   4452H
F2EF                VADRS: EQU   0F2EFH
                    ;
F2E0                PUTSUB:
F2E0 46                     LD    B,(HL)
F2E1 23                     INC   HL
F2E2 4E                     LD    C,(HL)
F2E3 C5                     PUSH  BC              ; B=CHR.CODE  C=ATR.CODE
F2E4 2AEFF2                 LD    HL,(VADRS)
F2E7 CD9D42                 CALL  VADDR           ; CALC VRAM.ADRS
F2EA C1                     POP   BC
F2EB CD5043                 CALL  VRMPUT
F2EE C9                     RET
                    ;
F2EF 00                     DB    0
                    ;
F2F0                GETSUB:
F2F0 E5                     PUSH  HL
F2F1 7E                     LD    A,(HL)
F2F2 23                     INC   HL
F2F3 66                     LD    H,(HL)
F2F4 6F                     LD    L,A
F2F5 CD5244                 CALL  VRMGET
F2F8 E1                     POP   HL
F2F9 77                     LD    (HL),A
F2FA 23                     INC   HL
F2FB 71                     LD    (HL),C
F2FC C9                     RET
                    ;
F2FD                        END
```

5. グラフィックデータ 書き込みサブルーチン

```
                ;-------------------------
                ; PC-TechKnow 8800 VOL.1
                ;  <<  GRAPHIC SUB  >>
                ;  (C) 1982 by Radix
                ;-------------------------
                ;
                ;  [[[ RELOCATABLE ]]]
                ;
                ;
                        ORG   866AH
                ;
F260            DATAAD:EQU   0F260H
                ;
866A 00                 DB    0
866B            START:
866B 7E                 LD    A,(HL)
866C 23                 INC   HL
866D 66                 LD    H,(HL)
866E 6F                 LD    L,A          ; HL = GVRAM.ADRS
                ;
866F ED5B60F2           LD    DE,(DATAAD)
                ;
8673 EB                 EX    DE,HL
8674 4E                 LD    C,(HL)
8675 23                 INC   HL
8676 46                 LD    B,(HL)
8677 23                 INC   HL
8678 EB                 EX    DE,HL        ; DE = DATA.POINTER
8679            GS0:
8679 C5                 PUSH  BC           ; BC = DATA.SIZE
867A            GS1:
867A F3                 DI
867B 1A                 LD    A,(DE)
867C D35C               OUT   (5CH),A
867E 77                 LD    (HL),A
867F D35F               OUT   (5FH),A
8681 13                 INC   DE
8682 1A                 LD    A,(DE)
8683 D35D               OUT   (5DH),A
8685 77                 LD    (HL),A
8686 D35F               OUT   (5FH),A
8688 13                 INC   DE
8689 1A                 LD    A,(DE)
868A D35E               OUT   (5EH),A
868C 77                 LD    (HL),A
868D D35F               OUT   (5FH),A
868F 13                 INC   DE
8690 FB                 EI
                ;
8691 23                 INC   HL
8692 10E6               DJNZ  GS1
                ;
8694 C1                 POP   BC
8695 C5                 PUSH  BC
8696 3E50               LD    A,80
8698 90                 SUB   B
8699 4F                 LD    C,A
869A 0600               LD    B,0
869C 09                 ADD   HL,BC        ; HL=HL+(80-B)
869D C1                 POP   BC
869E 0D                 DEC   C
869F 20D8               JR    NZ,GS0
86A1 C9                 RET
                ;
86A2                    END
```

6. 高速画面クリア

```
                  ;-------------------------
                  ; PC-TechKnow 8800 VOL.1
                  ;  <<  H.S. CLS 2  >>
                  ;  (C) 1982 by Radix
                  ;-------------------------
                  ;
                  ;  [[[ RELOCATABLE ]]]
                  ;
                         ORG  0866AH
                  ;
866A 00                  DB   0
866B             HSCLS2:
866B F3                  DI
866C D9                  EXX
866D 3E5C                LD   A,5CH
866F             LOOP:
866F 4F                  LD   C,A
8670 ED79                OUT  (C),A
8672 2100C0              LD   HL,0C000H
8675 1101C0              LD   DE,0C001H
8678 017F3E              LD   BC,15999
867B 3600                LD   (HL),0
867D EDB0                LDIR
867F 3C                  INC  A
8680 FE5F                CP   5FH
8682 20EB                JR   NZ,LOOP
                  ;
8684 D35F                OUT  (5FH),A
8686 D9                  EXX
8687 FB                  EI
8688 C9                  RET
                  ;
8689                     END
```

7.　カラーパレット・イニシャライズ

```
                    ;-------------------------
                    ; PC-TechKnow 8800 VOL.1
                    ;  <<  C.P. INIT >>
                    ;  (C) 1982 by Radix
                    ;-------------------------
                    ;
                    ;  [[[ RELOCATABLE ]]]
                    ;
                              ORG   0F320H
                    ;
F320                CPINIT:
F320 0608                     LD    B,8
F322 0E5B                     LD    C,5BH
F324                LOOP:
F324 05                       DEC   B
F325 ED41                     OUT   (C),B
F327 04                       INC   B
F328 0D                       DEC   C
F329 10F9                     DJNZ  LOOP
                    ;
F32B C9                       RET
                    ;
F32C                          END
```

8. サウンド・サブルーチン

```
                    ;------------------------
                    ; PC-TechKnow 8800 VOL.1
                    ;  << SOUND SUBROUTINE>>
                    ;  (C) 1982 by Radix
                    ;------------------------
                    ;
                            ORG   0E500H
                    ;
0393                SNERR:  EQU   0393H
3ECD                OUTSET:EQU    03ECDH
40BD                PTRLDH:EQU    040BDH
56FC                GETSTR:EQU    056FCH
E6C1                COPY:   EQU   0E6C1H
                    ;
E500 CDFC56                 CALL  GETSTR          ; Type check and Get string
E503 CDBD40                 CALL  PTRLDH          ; Load pointer
E506 57                     LD    D,A
E507 3E03                   LD    A,3             ; Default
E509 14                     INC   D
                    ;
E50A                SETLEN:
E50A 3C                     INC   A
E50B 5F                     LD    E,A
E50C                CHKEND:
E50C 15                     DEC   D               ; Check end
E50D C8                     RET   Z
                    ;
E50E 7E                     LD    A,(HL)          ; Data load
E50F 23                     INC   HL
E510 D631                   SUB   31H
E512 FE09                   CP    09H             ; Check numeral
E514 38F4                   JR    C,SETLEN
E516 D610                   SUB   10H
E518 E6DF                   AND   0DFH            ; Capital set
E51A FE1A                   CP    1AH             ; Check alphabet
E51C D29303                 JP    NC,SNERR
E51F E5                     PUSH  HL
E520 D5                     PUSH  DE
E521 2142E5                 LD    HL,TABL
E524 87                     ADD   A,A
E525 4F                     LD    C,A
E526 0600                   LD    B,0
E528 09                     ADD   HL,BC           ; Computation tabl address
E529 4E                     LD    C,(HL)
E52A 23                     INC   HL
E52B                UNITSN:
E52B 56                     LD    D,(HL)          ; Load time data
E52C                SOUND:
E52C F3                     DI
E52D 3AC1E6                 LD    A,(COPY)
E530 EE40                   XOR   40H             ; Negate bit6 at port 40H
E532 CDCD3E                 CALL  OUTSET
E535 41                     LD    B,C
E536                DLAY:
E536 10FE                   DJNZ  DLAY
                    ;
E538 15                     DEC   D               ; Count dowu time counter
E539 20F1                   JR    NZ,SOUND
                    ;
E53B 1D                     DEC   E               ; Count dowu length counter
E53C 20ED                   JR    NZ,UNITSN
                    ;
E53E D1                     POP   DE
E53F E1                     POP   HL
```

```
E540 18CA             JR   CHKEND
                ;
**** Interval and time data ****
E542            TABL:
E542 F82EEA32         DB   0F8H,02EH,0EAH,032H,0DDH,034H,0D0H,038H
E546 DD34D038
E54A C43AB83E         DB   0C4H,03AH,0B8H,03EH,0AEH,042H,0A4H,046H
E54E AE42A446
E552 9A4A914E         DB   09AH,04AH,091H,04EH,088H,054H,080H,058H
E556 88548058
E55A 795E7264         DB   079H,05EH,072H,064H,06BH,06AH,065H,070H
E55E 6B6A6570
E562 5F76597E         DB   05FH,076H,059H,07EH,053H,086H,04EH,08EH
E566 53864E8E
E56A 4A94459E         DB   04AH,094H,045H,09EH,041H,0A8H,03DH,0B2H
E56E 41A83DB2
E572 39BC36C6         DB   039H,0BCH,036H,0C6H
```

9.　ファンクションキー・イニシャライズ

```
                ;------------------------
                ; PC-TechKnow 8800 VOL.1
                ; << FUNCTION KEY INIT >>
                ;     (C) 1982 by Radix
                ;------------------------
                ;
                        ORG   0F260H
                ;
01B0            FKDATA:EQU   01B0H
3F79            DSFKEY:EQU   3F79H
                ;
E6F2            STRTAB:EQU   0E6F2H
                ;
00A0            LENFKT:EQU   0A0H

F260 E5                 PUSH HL
F261 21B001             LD   HL,FKDATA
F264 11F2E6             LD   DE,STRTAB
F267 01A000             LD   BC,LENFKT
F26A EDB0               LDIR
F26C CD793F             CALL DSFKEY
F26F E1                 POP  HL
F270 C9                 RET
                ;
F271                    END
```

10. PRINT to LPRINT

```
                ;________________________
                ; PC-TechKnow 8800 VOL.1
                ;   << PRINT/LPRINT >>
                ;   (C) 1982 by Radix
                ;________________________
                ;
                ;     CMD ON/OFF
                ;
                ;   [[[ RELOCATABLE ]]]
                ;
                ;
                        ORG   0F260H
                ;
0095            ON:     EQU   95H
00EE            OFF:    EQU   0EEH
                ;
0393            SNERR:  EQU   0393H
E64C            PRTFLG:EQU   0E64CH
                ;
                ;----- CMD COMMAND ( from 0EEB6H )
                ;
F260            CMD:
F260 FE95               CP    ON
F262 280C               JR    Z,CMD1
                ;
F264 FEEE               CP    OFF
F266 C29303             JP    NZ,SNERR
                ;
F269 D7                 RST   10H
F26A C29303             JP    NZ,SNERR
F26D AF                 XOR   A
F26E 1806               JR    CMD2
F270            CMD1:
F270 D7                 RST   10H
F271 C29303             JP    NZ,SNERR
F274 3E01               LD    A,1
F276            CMD2:
F276 3286F2             LD    (PLPFLG),A
F279 C9                 RET
                ;
                ;----- TOP OF PRINT ( from 0EDCFH )
                ;
F27A            PRNTOP:
F27A F5                 PUSH  AF
F27B 3A86F2             LD    A,(PLPFLG)
F27E B7                 OR    A
F27F 2803               JR    Z,NOTPLP
                ;
F281 324CE6             LD    (PRTFLG),A
F284            NOTPLP:
F284 F1                 POP   AF
F285 C9                 RET
                ;
                ;----- WORKING AREA
                ;
F286 00         PLPFLG:DB     0
                ;
F287                    END
```

11.　8インチ・フォーマット

```
                ;------------------------
                ; PC-TechKnow 8800 VOL.1
                ; < STADARD DISK FORMAT >
                ;  (C) 1982 by Radix
                ;------------------------
                ;
                ;  USR(<CYLINDER#>)   <CYLINDER#> must be INTEGER.
                ;  IF USR<>0 THEN ERROR
                ;
                        ORG  0E400H
                ;
3AF7            STSCK:  EQU  3AF7H
3C71            IWAIT:  EQU  3C71H
3C94            WTFDC:  EQU  3C94H
                ;
EC41            FACLO:  EQU  0EC41H
EF26            ST0:    EQU  0EF26H
EF27            ST1:    EQU  0EF27H
EF3E            WAITF:  EQU  0EF3EH
                ;
E400            FORMAT:
E400 7E                 LD   A,(HL)          ;CYLINDER#
E401 32F6E4             LD   (TRACK),A
E404 3E08               LD   A,08
E406 D3F5               OUT  (0F5H),A
E408 3E00               LD   A,0
E40A 32F5E4             LD   (HEAD),A
E40D            NEXT:
                ;--- MAKE TABLES         C,H,R,N *26
E40D 2111E5             LD   HL,TABLE
E410 061A               LD   B,26
E412 11F7E4             LD   DE,SECSQU
E415 3AF5E4             LD   A,(HEAD)
E418 0F                 RRCA
E419 0F                 RRCA
E41A 4F                 LD   C,A
                ;
E41B            MTBL1:
E41B 3AF6E4             LD   A,(TRACK)
E41E 77                 LD   (HL),A          ;C
E41F 23                 INC  HL
E420 71                 LD   (HL),C          ;H
E421 23                 INC  HL
E422 1A                 LD   A,(DE)
E423 77                 LD   (HL),A          ;R
E424 23                 INC  HL
E425 3E01               LD   A,1
E427 77                 LD   (HL),A          ;N
E428 23                 INC  HL
E429 13                 INC  DE
E42A 10EF               DJNZ MTBL1
                ;
                ;---SEEK HEAD
E42C 3E07               LD   A,7
E42E 323EEF             LD   (WAITF),A
E431 3E0F               LD   A,0FH           ;SEEK CMD
E433 CD943C             CALL WTFDC
E436 3AF5E4             LD   A,(HEAD)
E439 F601               OR   1               ;DRIVE=1
E43B CD943C             CALL WTFDC
E43E 3AF6E4             LD   A,(TRACK)
E441 CD943C             CALL WTFDC
E444 CD713C             CALL IWAIT
E447 3E00               LD   A,0
```

```
E449 323DEF           LD    (0EF3DH),A
E44C 3216EF           LD    (0EF16H),A
E44F 3A26EF           LD    A,(ST0)
E452 E6E0             AND   0E0H
E454 FE20             CP    20H
E456 C2E4E4           JP    NZ,ERR1
E459 3A27EF           LD    A,(ST1)
E45C 47               LD    B,A
E45D 3AF6E4           LD    A,(TRACK)
E460 B8               CP    B
E461 C2E7E4           JP    NZ,ERR2
               ;--- DMAC SET
E464 3E11             LD    A,TABLE
E466 D362             OUT   (62H),A
E468 3EE5             LD    A,TABLE.R8
E46A D362             OUT   (62H),A
E46C 3E67             LD    A,103              ;26*4-1
E46E D363             OUT   (63H),A
E470 3E80             LD    A,80H
E472 D363             OUT   (63H),A            ;TC=103 ,WRITE to DISK
E474 3EA6             LD    A,0A6H
E476 D368             OUT   (68H),A            ;START!
               ;
E478 3E02             LD    A,2
E47A D3F3             OUT   (0F3H),A           ;TYPE 0 SELECT
               ;--- SETF4
E47C 3AF6E4           LD    A,(TRACK)
E47F FE26             CP    26H
E481 3E0F             LD    A,0FH
E483 3802             JR    C,SETF41
E485 3E3F             LD    A,3FH
E487 D3F4      SETF41:OUT   (0F4H),A
               ;
               ;--- SEND WRITE ID
E489 3EFF             LD    A,0FFH
E48B 323EEF           LD    (WAITF),A
E48E 3E4D             LD    A,4DH              ;WRITE ID CMD for 765
E490 CD943C           CALL  WTFDC
E493 3AF5E4           LD    A,(HEAD)
E496 F601             OR    1                  ;DRIVE=1
E498 CD943C           CALL  WTFDC
E49B 3E01             LD    A,1
E49D CD943C           CALL  WTFDC              ;N=1
E4A0 3E1A             LD    A,26
E4A2 CD943C           CALL  WTFDC              ;SC=1A
E4A5 3E36             LD    A,36H
E4A7 CD943C           CALL  WTFDC              ;GPL=36
E4AA 3E40             LD    A,´@´
E4AC CD943C           CALL  WTFDC              ;INIT ´@´
E4AF CD713C           CALL  IWAIT
               ;
E4B2 3EA5             LD    A,0A5H
E4B4 D368             OUT   (68H),A            ;DMA STOP
E4B6 3E00             LD    A,0
E4B8 D3F3             OUT   (0F3H),A           ;DESELECT
E4BA CDF73A           CALL  STSCK              ;RECEVE RESULT
E4BD 3A26EF           LD    A,(ST0)
E4C0 47               LD    B,A
E4C1 3AF5E4           LD    A,(HEAD)
E4C4 F601             OR    1                  ;DRIVE=1
E4C6 B8               CP    B
E4C7 C2EAE4           JP    NZ,ERR3
E4CA 3A27EF           LD    A,(ST1)
E4CD B7               OR    A
E4CE C2EDE4           JP    NZ,ERR4
               ;
E4D1 3AF5E4           LD    A,(HEAD)
E4D4 B7               OR    A
```

```
E4D5 3E04               LD    A,4
E4D7 32F5E4             LD    (HEAD),A
E4DA CA0DE4             JP    Z,NEXT
                ;
E4DD 210000             LD    HL,0
E4E0 2241EC             LD    (FACLO),HL
E4E3 C9                 RET
                ;
E4E4 2E01       ERR1:   LD    L,1
E4E6 01                 DB    1
E4E7 2E02       ERR2:   LD    L,2
E4E9 01                 DB    1
E4EA 2E03       ERR3:   LD    L,3
E4EC 01                 DB    1
E4ED 2E04       ERR4:   LD    L,4
E4EF 2600               LD    H,0
E4F1 2241EC             LD    (FACLO),HL
E4F4 C9                 RET
                ;
                ;----- WORK AREA ----
                ;
E4F5 00         HEAD:   DB    0
E4F6 00         TRACK:  DB    0
E4F7 010E020F   SECSQU:DB     1,14,2,15,3,16,4,17,5,18,6,19,7,20
E4FB 03100411
E4FF 05120613
E503 0714
E505 08150916           DB    8,21,9,22,10,23,11,24,12,25,13,26
E509 0A170B18
E50D 0C190D1A
E511            TABLE:  DS    105
                ;

E57A                    END
```

12. 漢字フォント読み出しサブルーチン

```
                 ;------------------------
                 ; PC-TecKnow 8800 VOL.1
                 ; << READ KANJI FONT >>
                 ;  (C) 1982 by Radix
                 ;------------------------

                         ORG  0F2E0H
                 ;
7261             KANJI2:EQU   7261H
                 ;
F2E0             RKFSUB:
F2E0 6E                  LD   L,(HL)
F2E1 23                  INC  HL
F2E2 66                  LD   H,(HL)
F2E3 7D                  LD   A,L
F2E4 EB                  EX   DE,HL          ; DE = KANJI.CODE
                 ;
F2E5 F3                  DI
F2E6 3EFE                LD   A,0FEH
F2E8 D371                OUT  (71H),A        ; SELECT ROM5
                 ;
F2EA CD6172              CALL KANJI2         ; GET KANJI FONT TO WORK
                 ;
F2ED 3EFF                LD   A,0FFH
F2EF D371                OUT  (71H),A        ; SELECT MAIN ROM
F2F1 FB                  EI
                 ;
F2F2 C9                  RET                 ; RETURN TO BASIC
                 ;
F2F3                     END
```

13. ROLL文サブルーチン

```
                ;------------------------
                ; PC-TechKnow 8800 VOL.1
                ;  << SCROLL COMMAND >>
                ;  (C) 1982 by Radix
                ;-------------------------
                ;
                ;  DUMMY = USR(SCROLL.BYTE)
                ;
                ;  [[[ RELOCATABLE ]]]
                ;

                       ORG   0BF80H

005C            GVRAM0:EQU   5CH          ; OUTPUT PORT FOR SELECT GVRAM0
005D            GVRAM1:EQU   5DH          ; OUTPUT PORT FOR SELECT GVRAM1
005F            MAINRM:EQU   5FH          ; OUTPUT PORT FOR SELECT MAIN RAM
C000            GVRTOP:EQU   0C000H       ; GRAPHIC VRAM TOP ADDRESS
3E80            GVRBYT:EQU   16000
FE80            GVREND:EQU   GVRTOP+GVRBYT

                ;----- GET PARAMETER

BF80 7E                LD    A,(HL)
BF81 23                INC   HL
BF82 66                LD    H,(HL)
BF83 6F                LD    L,A          ; HL = SCROLL.BYTES

                ;-----  SET PARAMETER

BF84 E5                PUSH HL
BF85 E5                PUSH HL
BF86 1100C0            LD    DE,GVRTOP
BF89 19                ADD   HL,DE
BF8A E3                EX    (SP),HL
BF8B E5                PUSH HL
BF8C 21803E            LD    HL,GVRBYT
BF8F C1                POP   BC
BF90 B7                OR    A
BF91 ED42              SBC   HL,BC
BF93 4D                LD    C,L
BF94 44                LD    B,H
BF95 E1                POP   HL

                ;----- ACTION [A]

BF96 E5                PUSH HL
BF97 D5                PUSH DE
BF98 C5                PUSH BC
                ;
BF99 F3                DI
BF9A D35C              OUT   (GVRAM0),A
BF9C EDB0              LDIR
BF9E D35F              OUT   (MAINRM),A
BFA0 FB                EI

                ;----- ACTION [B]

BFA1 E1                POP   HL
BFA2 C1                POP   BC
                ;
BFA3 C5                PUSH BC
BFA4 E5                PUSH HL
                ;
BFA5 1180FE            LD    DE,GVREND
```

```
BFA8 7C                 LD    A,H
BFA9 F6C0               OR    0C0H
BFAB 67                 LD    H,A
BFAC          ACTB:
BFAC F3                 DI
BFAD D35D               OUT   (GVRAM1),A
BFAF 0A                 LD    A,(BC)
BFB0 D35C               OUT   (GVRAM0),A
BFB2 77                 LD    (HL),A
BFB3 D35F               OUT   (MAINRM),A
BFB5 FB                 EI
              ;
BFB6 23                 INC   HL
BFB7 03                 INC   BC
BFB8 E7                 RST   20H           ; CP HL,DE
BFB9 20F1               JR    NZ,ACTB

              ;----- ACTION [C]

BFBB C1                 POP   BC
BFBC D1                 POP   DE
BFBD E1                 POP   HL
              ;
BFBE C5                 PUSH BC
              ;
BFBF F3                 DI
BFC0 D35D               OUT   (GVRAM1),A
BFC2 EDB0               LDIR
BFC4 D35F               OUT   (MAINRM),A
BFC6 FB                 EI

              ;----- ACTION [D]

BFC7 E1                 POP   HL
              ;
BFC8 7C                 LD    A,H
BFC9 F6C0               OR    0C0H
BFCB 67                 LD    H,A
              ;
BFCC 5D                 LD    E,L
BFCD 54                 LD    D,H
BFCE 13                 INC   DE
BFCF C1                 POP   BC
              ;
BFD0 F3                 DI
BFD1 D35D               OUT   (GVRAM1),A
BFD3 3600               LD    (HL),0
BFD5 EDB0               LDIR
BFD7 D35F               OUT   (MAINRM),A
BFD9 FB                 EI
              ;
BFDA C9                 RET
              ;
BFDB                    END
```

14. DMA-ON

```
                   ;-------------------------
                   ; PC-TechKnow 8800 VOL.1
                   ;    <<   DMA  ON   >>
                   ;    (C) 1982 by Radix
                   ;-------------------------
                   ;
                           ORG   0F260H
                   ;
6FD1               CRTINI:EQU   6FD1H
705B               LPAG20:EQU   705BH
7066               LPAG25:EQU   7066H
                   ;
EF88               LINCNT:EQU   0EF88H
EF89               LINWDT:EQU   0EF89H
                   ;
F260               DMAON:
F260 215B70                LD    HL,LPAG20
F263 3A88EF                LD    A,(LINCNT)
F266 FE15                  CP    21
F268 3803                  JR    C,DMAON1
F26A 216670                LD    HL,LPAG25
F26D 3A89EF        DMAON1:LD    A,(LINWDT)
F270 FE50                  CP    80
F272 F3                    DI
F273 CDD16F                CALL  CRTINI
F276 FB                    EI
F277 C9                    RET

F278                       END
```

15. N-BASIC 1200ボー

```
                ;-----------------------------
                ;  PC-TechKnow 8800 VOL.1
                ;  << 1200bps for N-Basic >>
                ;  (C) 1982 by Radix
                ;-----------------------------
                ;
                        ORG     0E9C5H
                ;
0BFD            INITRD:EQU      0BFDH
0C50            INITWR:EQU      0C50H
3BDF            SNERR: EQU      03BDFH
EA66            COPY:  EQU      0EA66H
F1B6            HOOKRD:EQU      0F1B6H
F1B9            HOOKWR:EQU      0F1B9H
                ;
E9C5 7E                 LD      A,(HL)          ; cmd routine
E9C6 D642               SUB     'B'
E9C8 200A               JR      NZ,NOTB
E9CA 3EC9               LD      A,0C9H
E9CC            SETOP:
E9CC 32B6F1             LD      (HOOKRD),A
E9CF 32B9F1             LD      (HOOKWR),A
E9D2 D7                 RST     10H
E9D3 C9                 RET
                ;
E9D4            NOTB:
E9D4 3C                 INC     A
E9D5 C2DF3B             JP      NZ,SNERR
E9D8 E5                 PUSH    HL
E9D9 21EAE9             LD      HL,READST
E9DC 22B7F1             LD      (HOOKRD+1),HL
E9DF 21F5E9             LD      HL,WRITST
E9E2 22BAF1             LD      (HOOKWR+1),HL
E9E5 3EC3               LD      A,0C3H
E9E7 E1                 POP     HL
E9E8 18E2               JR      SETOP
                ;
E9EA            READST:
E9EA F1                 POP     AF
E9EB 3A66EA             LD      A,(COPY)
E9EE E60F               AND     0FH
E9F0 F618               OR      18H             ; 1200bps mode
E9F2 C3FD0B             JP      INITRD
                ;
E9F5            WRITST:
E9F5 F1                 POP     AF
E9F6 3A66EA             LD      A,(COPY)
E9F9 E60F               AND     0FH
E9FB F61C               OR      1CH             ; 1200bps mode
E9FD C3500C             JP      INITWR
                ;
EA00                    END
```

付録2　N_{88}-ROM　BASICインタプリタ分析

〔N_{88}-ROM　BASIC〕アドレス0000～7FFF

アドレス	項　目　名	機　　　能
0000	リセット	
0008	シンタックスチェック	RST (or CALL) の後に置かれているキャラクタと (HL) とを比較した上，一致しなかったらSYNTAX ERROR，一致したら RST10H をしてもどる。
0010	テキストからの 1文字読み込み	テキストから1文字 or 数を読み込む。
0018	1文字出力	CRT または LPT に1文字出力をする。 PRTFLG (E64C) = 0　CRT ≠0　LPT
0020	HL，DE の比較	HL－DE（無符号）を行いフラグをセットする。 ACC はこわれる。
0028	SGN (FAC)	FAC（単精度，倍精度）の符号を調べる。 －：ACC =FF，NZ，M 0：　= 0，Z，P ＋：　= 1，NE，P
0030	FAC の型チェック	FAC の型を調べる。 INT：ACC = FF，C，NZ，M，PE STR：　= 0，C，Z，P，PE SNG：　= 1，C，NZ，P，PO DBL：　= 5，NC，NZ，P，PE
0038	ユーザ用 RST	モニタではブレークポイント用またはコマンド待ちにもどる時に使用する。
004A	ダイレクトモード	CURLIN (E656) に FFFF を入れる。
008A ∫ 0095	演算子優先順位 テーブル	+，－，＊，／，，AND，OR，XOR，EQV，IMP，MOD，¥の順に優先度を表わす数が入っている（数字が大きいほど、優先度が高い）。

アドレス	項 目 名	機 能
0096 ~ 009F	FAC 型変換ルーチン アドレス・テーブル	0096 ：to DBL 009A：to INT 009C：to STR 009E：to SNG
00A0 ~ 00A9	倍精度演算ルーチン アドレス・テーブル	＋，－，＊，／，比較の順 FAC ◦(EC4A~EC51) → FAC
00AA ~ 00B3	単精度演算ルーチン アドレス・テーブル	BCDE ◦ FAC → FAC
00B4 ~ 00BD	整数演算ルーチン アドレス・テーブル	DE ◦ HL → FAC オーバーフローがなければ HL にも入る。
00BE ~ 030D	ワークエリア初期データ	E600~に転送される。
030E	文字列	DB ′ Error′, 0
0315	文字列	DB ′ in ′, 0
0319	文字列	DB ′OK′, FF, 0D, 0A, 0
0320	文字列	DB ′Break′, 0
0326 ~ 0345	インタラプト・ベクトル	F300 ~に転送される。 (2バイト×16組)
0346	スタックサーチ	FOR, GOSUB 用のスタックを見つける。
034F	READYR	スタックを文実行前にもどしてコマンド待ちへ。
0375	プログラムエンド処理	プログラムエンドに来た時の処理ルーチン。
0393	SNERR	Syntax error 出力
0396	DUPERR	Duplicate label 出力
0399	LBLUND	Undefined label 出力
039C	DU0ERR	Division by zero 出力
039F	NFERR	Next without For 出力
03A2	DDERR	Duplicate Definition 出力
03A5	UFERR	Undefined User function 出力
03A8	REERR	Resume without error 出力
03AB	OVERR	Overflow 出力
03AE	MOERR	Missing operand 出力
03B1	TMERR	Type mismatch 出力
03B3	エラー出力	Ereg にエラーコードを入れてジャンプすると，エラーが出力される。

アドレス	項　目　名	機　　能
047B	READY	OK を表示しコマンド待ちになる。
04A7	MAIN	コマンド待ち。
05BD	LINKER	テキストのリンクポインタを正しくセットする。
05E9	ペア行番号GET	ペアの行番号を読み込む。
0605	行サーチ	テキスト中から行番号＝ DE の文を探す。
0632	中間言語変換	(HL) ～ 00 のテキストを中間言語に変換し，E87A ～に入れる。
08BF	FOR	FOR 文エントリ
0996	次の文を実行	
09C4	次の行を実行	
09E5	1文実行	テキスト　ポインタ＝ HL
0A0D	1文字読み込み	RST 10H とほぼ同じ。
0A8D	数→ FAC	RST 10H 用FAC →演算用 FAC
0AC4	DEFSTR	DEFSTR 文エントリ
0AC7	DEFINT	DEFINT 文エントリ
0ACA	DEFSNG	DEFSNG 文エントリ
0ACD	DEFDBL	DEFDBL 文エントリ
0B01	正の整数式の評価	正の整数式を評価して DE に入れる。
0B0B	行番号読み込み	行番号を読んで DE に入れる。
0B34	行位置読み込み	行番号 or 行アドレスを読み込んで DE に入れる。
0B7C	RUN	RUN エントリ
0BBF	GOSUB	GOSUB 文エントリ
0BE0	割込み GOSUB	ON 割込み発生時の GOSUB
0BF9	GOTO	GOTO 文エントリ
0C41	RETURN	RETURN 文エントリ
0C77	DATA	DATA 文エントリ
0C79	REM	REM 文エントリ
	ELSE	ELSE 文エントリ
0C97	FAC →変数	FAC の値を変数にコピーする。 Acc ←型－3 DE ←コピー先アドレス PUSH リターンアドレス PUSH DE PUSH AF JP 0C97
0C9C	LET	LET 文エントリ
0D01	ON	ON 文エントリ
0D05	ON ERROR	ON ERROR GOTO 処理
0D39	ON XX GOSUB	ON 〈割り込み〉 GOSUB （　）の処理

アドレス	項　目　名	機　　　能
0D73	ON〈式〉GOTO GOSUB	ON ～ GOTO GOSUB の処理
0D8D	RESUME	RESUME 文エントリ
0DCA	ERROR	ERROR 文エントリ
0DD5	AUTO	AUTO 文エントリ
0E05	IF	IF 文エントリ
0E4C	LPRINT	LPRINT 文エントリ
0E54	PRINT	PRINT 文エントリ
0E8F 〜 0ECD	文字列出力	FAC の文字列を位置制御して出力する処理を行っている。
0ED2	コンマ処理	PRINT 文での“,”の処理
0F1B	TAB，SPC 処理	PRINT 文での TAB，SPC の処理
0F8B	カセット OFF	カセットアクセスを終る。
0FAA	LINE	LINE 文エントリ
0FAF	LINE INPUT	LINE INPUT 処理
102D	INPUT	INPUT 文エントリ
1040 〜 1063	プロンプト文処理	(LINE) INPUT のプロンプト文の処理ルーチン。
10F9	READ	READ 文エントリ
11D3	式の評価 (FRMEUL)	式を計算して，結果を FAC に入れる。
1341	整数除算(1)	HL/DE → FAC
1350	因子の評価	演算の因子を評価して FAC に入れる。
1394	ERR	ERR 変数→ FAC
1398	Acc to FAC	Acc (0 ～255) を整数型 FAC に入れる。
13A2	ERL	ERL 変数→ FAC
13B0	VARPTR	VARPTR → FAC
13F2	(式) の評価	(　) で囲まれた式を評価して FAC に入れる。
1406	変数読み出し	変数の値→ FAC
1415	大文字変換	Acc 内が英小文字だったら大文字に変換する。
1512	NOT	
1581	LPOS	LPOS → FAC（LPOS の値＝ E64B 番地）
1586	POS	POS (X) → FAC
158F	USR	USR エントリ
15C8	DEF USR	DEF USR 処理
15D7	DEF	DEF エントリ
1600	FN	FN エントリ

アドレス	項　目　名	機　　　能
1796	FAC 型変換	FAC を Acc で示す型に変換する。 Acc ＝ 2 INT 3 STR 4 SNG 8 DBL
17A5	ブロック転送	(DE＋＋) → (HL＋＋), BC−−, repeat until BC ＝ 0
17AF	プログラム実行中チェック	ダイレクトモード中だったら Illeagal direct エラー。
17C9	FF グループ・ステートメント	FF グループのステートメントの各処理ルーチンへ分岐する。
17E5	INP	INP (FAC) → FAC
17FA	OUT	OUT 文エントリ
1800	WAIT	WAIT 文エントリ
181A	WIDTH	WIDTH 文エントリ
1856	WIDTH #	Acc ←ファイル#, E ← サイズにして CALL
186A	WIDTH LPRINT	Acc ← 文字数にして CALL
1880	WIDTH PRINT	PRINT 文での WIDTH 設定, Acc ← 文字数 & CALL
1896	整数式の評価	HL ←テキストポインタ & CALL FAC と DE に値が入る。
18A3	パラメータの評価 (GETBYT)	価が 0 ～255となる式の評価, HL ←テキストポインタ結果は Acc, Ereg, FAC に入る。
18D4	LLIST	LLIST エントリ
18D9	LIST	LIST エントリ
194C	LIST 形式変換	(HL)～ 0 0 の中間言語を LIST 形式に変換して入力バッファ (E9B9 ～)に入れる。
1B40	DELETE	DELETE エントリ
1B7A	PEEK	PEEK (FAC)→FAC
1B84	POKE	POKE エントリ
1B9D	FAC アドレス変換	FAC に入っている−32768～65535の数値を整数に直し, HL に入れる。
1BBA	行番号→アドレス	プログラム中の全部の GOTO, GOSUB などの行番号の
1BBB	アドレス→行番号	行番号←→行アドレス変換を行う。
1C89	OPTION BASE	OPTION エントリ
1CCD	コンソール出力	Acc の文字をコンソールへ出力する。59A4 と同じ Acc ← CHRコード & CALL
1CD1	RANDOMIZE	RANDOMIZE エントリ
1D2A	ループエンドサーチ	Creg ＝ 1A のとき, NEXTサーチ ≠ 1A のとき, WEND サーチ

アドレス	項目名	機能
1DDB	SNG FAC インクリメント	単精度FACをインクリメントする。
1DE6	単精度減算	BCDE − FAC → FAC
1DE9	単精度加算	BCDE ＋ FAC → FAC
1F10	LOG	LOG (FAC) → FAC
1F53	単精度乗算	BCDE × FAC → FAC
1FB7	単精度除算	BCDE ／ FAC → FAC
20A0	ABS	ABS (FAC) → FAC
20AB	NEG(FAC) (1)	単精度，倍精度FACの符号反転。
20B3	SGN	SGN (FAC)→FAC
20B6	Acc → FAC (CONIA)	Acc (−128 ～ 127) を整数型FACに入れる。
20BD	SGN (FAC)	FACの符号を調べる。 ＋：Acc ＝ 1，NZ, P 0： 0，Z，P −： FF, NZ, M
20CD	単精度 FAC PUSH (PUSH F)	単精度のFACをPUSHする。
20DA	MOVFN	(HL)～(HL＋3)→ SNG FAC
20DD	MOVFR	BCDE → SNG FAC
20E8	MOVRF	SNG FAC → BCDE
20EB	MOVRM	(HL)～(HL＋3)→ E, D, C, B
20ED	GETBCD	(HL)～(HL＋2)→ D, C, B
20F4	MOVMF	SNG FAC →(HL)～(HL＋3)
2107	フォーマット変換	FAC, BCDE記憶用フォーマット→演算用フォーマット
212B	FACLO GET	DE ←タイプ別FACアドレス
2134	単精度比較	BCDE − FACを行ってAcc, フラグをセットする。
215F	整数比較	DE − HLを行ってAcc, フラグをセットする。
2199	倍精度比較	FAC −(EC4A ～ EC51) を行ってAcc, フラグをセットする。
21A0	CINT (FRCINT)	FACを整数型に変換する。
21FD	MKINT	HL → INT FAC
2214	CSNG (FRCSNG)	FACを単精度に変換する。
223E	CDBL (FRCDBL)	FACを倍精度に変換する。
2256	ストリングチェック (CHKSTR)	FACが文字型であることのチェック。 (ちがうと Type mismatch)

アドレス	項　目　名	機　　能
2286	FIX	FIX (FAC)→ FAC
2295	INT	INT (FAC)→ FAC
231F	整数減算	DE − HL → FAC，HL
233A	整数加算	DE + HL → FAC，HL
235A	整数乗算	DE × HL → FAC，HL
23AB	整数除算(2)	DE ¥ HL → FAC，HL
23F7	NEG (FAC) (2)	整数型 FAC の符号反転
240C	整数剰余	DE mod HL → FAC，HL
241D	倍精度減算	FAC −(EC4A ~ EC51)→ FAC
2424	倍精度加算	FAC +(EC4A ~ EC51)→ FAC
2553	倍精度乗算	FAC ×(EC4A ~ EC51)→ FAC
2629	倍精度除算	FAC ／(EC4A ~ EC51)→ FAC
26B5	倍精度数入力	数字（外部形式）→ FAC，ポインタ HL
26BC	単精度数入力	〃
28D0	浮動小数点出力 (free format)	FAC → EC53 ~ に数字（外部形式）
28D1	浮動小数点出力 (fixed format)	FAC → EC53 ~に数字（外部形式） Breg ←整数部の桁数（小数点は含まない） Creg ←小数部の桁数（　　〃　　） Areg bit 7：1 bit 6：3ケタごとの区切り「,」を入れる／入れない bit 5：先頭を「*」で埋める／埋めない bit 4：$をつける／つけない bit 3：+をつける／つけない bit 2：符号は数字の後／前 bit 1：未使用 bit 0：指数形式
2E05	SQR（単精度）	SQR (FAC)→ SQR
2E15	FPWR（単精度）	BCDE ^ FAC → FAC
2E6E	EXP（単精度）	EXP (FAC)→ FAC
2ED8	多項式計算(1) （単精度）	$C_0 \times FAC + C_1 \times FAC^3 + C_2 \times FAC^5 + \ldots$ → FAC
2EE7	多項式計算(2) （単精度）	$C_0 + C_1 \times FAC + C_2 \times FAC^2 + C_3 \times FAC^3 + \ldots$ → FAC
2F1A	RND（単精度）	RND (FAC)→ FAC
2F8B	COS（単精度）	COS (FAC)→ FAC
2F91	SIN （単精度）	SIN (FAC)→ FAC
302C	TAN（単精度）	TAN (FAC)→ FAC

アドレス	項目名	機能
3041 ∫ 3052	パラメータテーブル 1	DMA タイプ 5インチディスク用パラメータ
3053 ∫ 3064	パラメータテーブル 2	DMA タイプ 8インチディスク用パラメータ
3065 ∫ 3069	シフトテーブル	シフト用データ
306A ∫ 3071	RHT MSK	
3072 ∫ 3078	LFT MSK	
3079	INT BAD	末使用インタラプト用ルーチン，何もせずにもどる。
3080	VRTC インタラプト	VRTC インタラプト・ルーチン PORT E6H，bit 1 ← 0 でマスクできる。
3167	RS-232C インタラプト	RS-232C 入力インタラプト・ルーチン PORT E6H，bit 2 ← 0 でマスクできる。
3203	カナ出力	RS-232C ポートに SI/SO プロトコルでカナを出力する。
3242	キースキャン	キースキャンを行う。 CY = 1 キー押されてない。 = 0，Z = 1 キー押したまま } EFFE にデータ Z = 0 新しいキーを押した
3583	キー入力	キーボード（正確にはキー入力用キュー）から1文字入力して Acc に入れる。入力待ちあり。
35A8	COPY 処理	COPY キーの処理
35C2	ブレークチェック	STOP，^C のときキャリーをセットしてもどる。
35CE	キー入力	キーバッファから1文字読む。 Z：バッファエンプティ NZ：Acc ←入力データ
35D9	キーバッファイニシャライズ	キー入力用キューをイニシャライズする。
369A	DISK I/O (PHYDIO)	物理的 DISK I/O ルーチン entry : EC85 ←ドライブ♯（0～） EF50 ←ドライブタイプ 8インチ DMAタイプ………… 0

アドレス	項　目　名	機　　　能
		5インチDMAタイプ…………1 5インチ片面インテリジェント‥2 5インチ両面インテリジェント‥3 ECB4 ← 0（エラーカウント） HL ← データアドレス BC ← トラック#，セクタ# フラグ ← I/O モード CY = 1……ライト CY = 0，Z……チェック CY = 0，NZ……リード exit : CY = 1：エラー = 0：正常終了、BCHLA は保存
36E2	PC-8031 (- 2W) イニシャライズ	PC-8031 (- 2W) をイニシャライズする。 exit : Acc ←ドライブ数
37C9	コマンド送出	PC-8031 にコマンドバイトを送る。 Acc ←コマンド & CALL
37D2	データ送出	PC-8031 (- 2W)にデータバイトを送る。
3847	データ受信	PC-8031 (- 2W)からデータを受けとる。 Acc にデータが入る。CY = 1 の時はエラー
3A88	HEAD RESTORE	DMA タイプドライブのヘッドリストア。 Creg ←××××× HD US1 US0 & CALL
3AAD	SEEK TRACK	DMA タイプドライブのトラックシーク Creg ← ××××× HD US1 US0 Dreg ←トラック#　してCALL
3AF7	STATUS CHECK	DMA タイプドライブのステータス読み込み。
3C71	インタラプト待ち	FDC からのインタラプトを待つ。
3C7F	リード FDC	FDC からデータを読む。
3C94	ライト FDC	FDC へデータを書き込む。
3CAC	DMA 8インタラプト	8インチ DMA タイプ インタラプトルーチン
3C39	DMA 5インタラプト	5インチ　〃
3DCB	ドライブタイプゲット	entry : Acc ←ドライブ#－1 exit : Acc，EF5D =ドライブタイプ
3E0D	CRT 出力	CRT への1文字出力（Acc=ASCIIコード）
3E9B	BELL	BEEP 出力
3EB4	BEEP	BEEP エントリ
3ED4	プリンタ出力ハンドラ	プリンタへ1バイト送る。(Acc=ASCIIコード)
3F27	プリンタ ESC+出力	プリンタ ESC + Acc を送る。
3F31	CSRLIN	CSRLIN → FAC

アドレス	項目名	機能
3F3F	PEN 割込みチェック	PEN 割込みを起こすかどうかのチェック
3F62	PEN リード	ライトペン位置読み込み　H, L ← X, Y
3F79	ファンクションキー表示	ファンクションキー（f・1 ～ f・5）の表示 (E6B8)＝0 のとき表示しない。 ≠0 のとき表示する。
3F7A	ファンクションキー表示	ファンクションキーの表示（E6B8 による） Acc ＝ 0 のとき f・1 ～ f・5 Acc ＝ 5 のとき f・6 ～ f・10 を表示する。
4015	ON インタラプトベクトル アドレス GET	entry : Acc ← ON インタラプト# exit : HL ←インタラプトベクトルアドレス
4021	最下行クリア	テキスト画面の最下行をクリアする。
4047	タイマリード	タイマ→タイムバッファ (F00D ～ F011)
4121	INPUT WAIT 処理	(LINE) INPUT WAIT ◯の WAIT の処理をする。
4143	CLOCK インタラプト	CLOCK インタラプトルーチン
4242	クロック割込み禁止	クロック割込みを禁止する。
4250	〃　許可	〃　許可する。
4257 ｛ 428A	VRAM アドレス	VRAM 上の各行の先頭アドレス－ VRAMTOP の値が格納されている。
428B	カーソル OFF	カーソルを表示しない。
4290	カーソル ON	カーソルを表示する。
429D	VRAM アドレス計算	カーソル位置 (H, L)→ カーソルアドレス (HL) カーソル位置は 1 ～
42B4	ヌルライン イニシャライズ	ヌルライン（通常 FF80 ～ FFF7）をイニシャライズする。
42D4	UP スクロール	テキスト画面を UP スクロールする。
42FE	DOWN スクロール	テキスト画面を DOWN スクロールする。
431E	ラインアドレス	entry : L ←テキスト画面　行#(1 ～) exit : DE ＝行の先頭のアドレス
433F	VRT 同期	Vertical retrace になるまで待つ。
434C	VRAM 1 文字 PUT	B ←キャラクタコード HL ← VRAM 上アドレス　& CALL
4350	〃	B ←キャラクタコード C ←アトリビュートコード HL ← VRAM 上アドレス　& CALL
4351	アトリビュートセット	C ←アトリビュートコード。 HL ← VRAM 上アドレス & CALL

アドレス	項　目　名	機　　　　能
4452	VRAM 1文字 GET	entry : HL ← VRAM 上アドレス exit :　Acc, B ＝キャラクタコード 　　　　　C ＝アトリビュートコード
445B	アトリビュート作成	entry : Acc ←ファンクションコード（カラーコード） exit :　Acc ＝アトリビュートコード
4472	スクリーン 1 文字 GET	entry : H, L ← X, Y（1～） exit :　Acc, B ＝キャラクタコード 　　　　　C ＝アトリビュートコード
447D	スクリーン 1 文字 PUT	entry : EF86, 7 ← X, Y（1～） 　　　　Acc ＝ キャラクタコード
449F	POS (X)	POS (X)→ Acc
44A4	アドレス変換	実アドレス (HL)→ ウィンドウ内アドレス (HL)
44B3	アドレス変換	実アドレス (BC)→ ウィンドウ内アドレス (BC)
44D5	アドレス変換	ウィンドウ内アドレス (HL)→ 実アドレス (HL)
4551	ROM 5 CALL	ROM 5 の処理を CALL する。
458F	IPL ロード	IPL をロードし，ジャンプする。
45DD	PUT キュー	キューにデータを入れる。 　Acc ←キュー＃，E ←データ & CALL
45F8	GET キュー	キューからデータを読み出す。 　entry : Acc ←キュー＃ (0 or 1) 　exit :　　Z ：キューエンプティ 　　　　　NZ：Acc ＝データ
461B	POP キュー	PUT キューを行わなかったことにする。 　entry : Acc ←キュー＃ exit :　　Z ：キューエンプティ 　　　　　NZ：Acc ＝ PUT したデータ
463D	キュー イニシャライズ	キューのイニシャライズをする。 　entry : Acc ←キュー＃ 　　　　　B　←キュー長 (2^n-1) 　　　　　DE ←キューアドレス
464E	BACK キュー	データをキューの TOP に入れる。 　entry : Acc ←キュー＃ 　　　　　E　←データ
4655	キュー・データ長	キューに入っているデータの数を HL に入れてもどる。
4667	FRE キュー	キューの空いているバイト数を HL, Acc に入れてもどる。
4680	キュー・テーブル・アドレス	entry : Acc ←キュー＃ exit :　HL ←キュー・テーブル・アドレス
468C	ファイル・ディスクリプタ処理	ファイル・ディスクリプタの処理をする。

アドレス	項 目 名	機 能
46F8	VARPTR (#n)	#Acc の VARPTR を HL に入れる。
4798	OPEN	OPEN エントリ
481D	ファイル CLOSE	#Acc のファイルを CLOSE する。
4852	RUN “ ”	RUN 〈ファイル〉 エントリ
4854	LOAD	LOAD エントリ
4855	MERGE	MERGE エントリ
49AA	RSET	RSET エントリ
49AB	LSET	LSET エントリ
4A5C	FIELD	FIELD エントリ
4AA1	MKI$	
4AA4	MKS$	MK○$ (FAC) → FAC
4AA7	MKD$	
4ABA	CVI	
4ABD	CVS	CV○ (FAC)→ FAC
4AC0	CVD	
4B04	CLOSE	CLOSE エントリ
4B0C	CLOSE ALL	すべてのファイルを CLOSE する。
4B41	PUT #n	PUT #n 処理ルーチン
4B42	GET #n	GET #n 処理ルーチン
4B54	ファイル出力	カレントファイルにデータを出力する。
4B7B	ファイル入力	カレントファイルからのデータのシーケンシャル入力。
4BAC	INPUT$	INPUT$ エントリ
4C2F	LOC	LOC (FAC)→ FAC
4C40	LOF	LOF (FAC)→ FAC
4C51	EOF	EOF (FAC)→ FAC
4C62	FPOS	FPOS (FAC)→ FAC
4CC1	LINE INPUT#	LINE INPUT# 処理ルーチン
4DA0		Bad file name
4DA3		File already open
4DA6		Direct statement in f ile
4DA9		File not found
4DAC		File not open
4DAF		FIELD overflow
4DB2		Bad f ile number
4DB5		Internal error
4DB8		Input past end
4DBB		Sequential after PUT
4DBE		Sequential I/O only
4DC1		Feature not available

アドレス	項　目　名	機　　　能
4E53 〜 4E7B	デバイス名 デバイス＃　テーブル	デバイス名とそのデバイス＃が入っている。
4E7C 〜 4E8B	処理ルーチンテーブル アドレス	デバイスごとの I/O 処理ルーチンテーブルの先頭アドレスが入っている。
4E8C	シーケンシャルデバイス各種処理 (GENDSP)	DISK 以外のデバイスに対する各処理ルーチンに分岐する。 entry : Acc ←処理＃×2 HL ← VARPTR (＃n) D ←デバイス＃ その他処理により他のレジスタ，フラグ，スタックにパラメータが必要 処理＃　処理 0　OPEN 1　CLOSE 2　PUT/GET 3　OUTPUT 4　INPUT 5　LOC 6　LOF 7　EOF 8　FPOS 9　BACK READ DATA 10　WIDTH
4EC1	OUT of memory チェック	C ←必要なスタックレベル & CALL
4F01	NEW	NEW を行なう。
4F21	CLEAR	変数の CLEAR を行なう。
4FE5	ON インタラプト ON	ON インタラプトを ON にする。
4FF5	ON インタラプト OFF	〃　OFF にする。
4FFB	ON インタラプト STOP	〃　STOP にする。
5012	ON インタラプト リクエスト	〃　リクエストする。 entry : HL ←割り込みベクトルアドレス (cf 4015)
5045	ALL OFF	すべての ON インタラプトを OFF にする。
5059	ポーリング	ON インタラプトのポーリングをして GOSUB する。
50A5	RESTORE	RESTORE エントリ

アドレス	項　目　名	機　　　　能
50CA	STOP	STOP エントリ
50E5	END	END エントリ
5140	CONT	CONT エントリ
5158	TRON	TRON エントリ　TRON フラグ= EC3B
5159	TROFF	TROFF エントリ
515E	SWAP	SWAP エントリ
519C	ERASE	ERASE エントリ
522E	CLEAR	CLEAR エントリ
52BD	NEXT	NEXT エントリ
537D	ラベルサーチ	ラベルテーブルからラベルのサーチをする。
53AF	ラベル長カウント	ラベルの長さを数える。
53F6	ラベル登録	プログラム中のラベルをすべて登録する。
5441	ラベルリファレンス	ラベル名→ラベルのついている行のアドレス entry : HL ←テキストポインタ（＊の所） exit :　HL ←テキストポインタ（ラベルの次） DE ←ラベルのついている行のアドレス
5480	ラベルスキップ	
5494	ストリング 比較	
54C1	OCT$	OCT$ (FAC)→ FAC
54C6	HEX$	HEX$ (FAC)→ FAC
54CB	STR$	STR$ (FAC)→ FAC
54D8	ストリングコピー	文字列領域にコピーの文字列をつくる。 HL ←ソース ストリング ディスクリプタ アドレスをしてCALLする。 exit : DE ←コピーストリング ディスクリプタ アドレス
54EE	ストリングスペース確保	Acc 個のストリングスペースを確保する。 HL に そのディスクリプタのアドレスが入る。
5500	ストリング登録	(HL ＋ 1)～ 00 or Breg or Dreg をストリングとしてストリングスタックに PUSH し，FAC に入れる。
5530	ストリング登録	DE が指すディスクリプタの文字列をストリングスタックに PUSH し、FAC に入れる。
5550	文字列出力	(HL)～ 00 の文字列を出力する。
5572	ストリングスペース確保	Acc 個のストリングスペースを確保する。 DE に そのアドレスが入る。ディスクリプタは作られない。
559A	ガーベジコレクション	ガーベジコレクションを行なう。
56BA	ストリングコピー	HL ←ソースストリングのディスクリプタアドレス DE ←ディスティネーションスペースのアドレス
56CC	ストリング POP	FACのストリングをストリングスタックからPOPする。

アドレス	項 目 名	機 能
5 6 F 8	LEN	LEN (FAC)→ FAC
5 6 FC		→ Acc
5 7 0 4	ASC	ASC (FAC)→ FAC
5 7 0 8		→ Acc
5 7 1 4	CHR$	CHR$ (FAC)→ FAC
5 7 2 2	STRING$	STRING$ エントリ
5 7 4 1	SPACE$	SPACE$ (FAC)→ FAC
5 7 5 A	LEFT$	LEFT$ エントリ
5 7 8 A	RIGHT$	RIGHT$ エントリ
5 7 9 3	MID$	MID$ 関数エントリ
5 7 B 4	VAL	VAL (FAC)→ FAC
5 7 D 7	INSTR	INSTR エントリ
5 8 1 6	INSTR	INSTR (P, A$, B$)→ Acc entry : Acc ← P DE ← VARPTR (B$) HL ← VARPTR (A$) exit : Acc ← INSTR
5 8 5 A	MID$ 文	MID$ 文エントリ
5 8 E 4	FRE	FRE (FAC)→ FAC
5 9 3 5	プリンタ出力	プリンタへ１文字出力する。
5 9 8 9	プリンタ改行	LPOS ≠ 0 のときプリンタを改行する。
5 9 A 4	コンソール出力	コンソールへ１文字出力する。
5 A 0 8	１文字入力	入力デバイスから１文字入力する。
5 A 5 8	改行	POS (X)≠ 0 のとき改行する。
5 A 8 6	イベントキーチェック	イベントキー（^O，^S，^C）が押されていたら，それに応じた動作を行う。
5 AA 3	INKEY$	INKEY$ エントリ
5 AA 4		INKEY$ → FAC
5 AC 5	DIM	DIM エントリ
5 ACA	変数アドレス GET	entry : HL ←テキストポインタ（変数名の最初） exit : DE ← VARPTR
5 DD 5 ∫ 5 DE 4	カーソル移動コード 処理アドレステーブル	LF，HOME，CLR，CR，⇥，⇤，⇧，⇩の順 カーソル移動処理を行うルーチンのアドレステーブル。
5 DE 5	カーソル進める	⇥
5 DF 7	カーソル復帰	CR
5 E 0 3	カーソル改行	LF
5 E 2 3	カーソル DOWN	⇩
5 E 3 3	カーソル UP	⇧

アドレス	項目名	機能
5E49	ホームポジション	HOME
5E54	カーソル後退	⇤
5F0E	画面クリア	CLR
5F76	行継続コード読み込み	L 行（1〜）が次の行とつながっているかどうかを示すコードを読み込む。 cf. EF9A 〜
5F86	行継続コード書き込み	行継続コードを書き込む。
5F92	1行入力 (1)	カーソルのある行全部を読み込む。
5FC8	1行入力 (2)	カーソル位置から入力した所までを読み込む。 exit : E9B9 〜 入力文字列 CY = 1: STOP, ^C = 0: ☑
657B	EDIT	EDIT エントリ
6642	PRINT USING	PRINT USING 処理
67EF	CSAVE	カセットへのプログラムセーブを行う。 F009 = FBH 1200ボー = FAH 600ボー
680F	CLOAD	カセットからのロード，ベリファイ Acc = 0 ロード = FF ベリファイ
69B2	フックイニシャライズ	フックをイニシャライズする。
69D1	インタラプトベクトルイニシャライズ	インタラプトベクトル (F300 〜 F31F) をイニシャライズする。
69E1	CRT イニシャライズ	CRT をイニシャライズする。
69FE 〜 6AAF	処理ルーチンアドレステーブル	中間言語 81 H 〜 D9 H に対する処理ルーチンのアドレステーブル。
6AB0 〜 6B01	FF シリーズ前半の処理ルーチンアドレステーブル	中間言語 FF81 〜 FFA9H の関数に対する処理ルーチンのアドレステーブル。
6B02 〜 6B55	FF シリーズ後半の処理ルーチンアドレステーブル	中間言語 FFD0 〜 FFE4H に対する関数・文としての処理アドレステーブル。 4バイトで1組 前の2バイト……関数としての処理ルーチンアドレス 後の2バイト……文 〃
6B56 〜 6B89	予約語のインデックス	予約語テーブルの最初の1文字による INDEX

アドレス	項　目　名	機　　　能
6B8A ～ 6E80	予約語テーブル	予約語とその中間言語のリストが入っている。
6E81 ～ 6E95	演算子テーブル	1文字の予約語とその中間言語のリスト。
6E96 ～ 6F11	ROM 5 処理呼び出し	ROM 5 処理を呼び出すために使われる。4バイトで1組 CALL　4551 H DB　　〈処理#〉 これを CALL することにより〈処理#〉に対応する処理が行われる。ROM 5 ルーチン一覧表参照
6F12 ～ 6F69	エラーメッセージ	ROM-BASIC のエラーメッセージテーブル
6F6B	テキスト画面 WIDTH	テキスト画面のイニシャライズ entry : B ←桁数 C ←行数 その他の情報は，ワークエリアに従ってセットされる。
6FD1	CRTC セット	CRTC，DMAC をセットする。 entry : CY = 1　40ケタ，CY = 0　80ケタ (E6B9)= FF（カラー），0（モノクロ） (E6C4, 5)= VRAM TOP アドレス (F3C8) HL = 705BH（20行），7066H（25行）
7071	CONSOLE	CONSOLE エントリ
714E	LOCATE	LOCATE エントリ
7198	GET	GET エントリ
71A6	PUT	PUT エントリ
71B5	CLS	CLS エントリ
71C6	CLS	B ←〈機能〉& CALL
71D9	タイマセット	タイムバッファ (F00D ～ F011)→ タイマ
721C	DATE$	DATE$ 文エントリ
7279	TIME$	TIME$ 文エントリ
72AB	HELP	HELP 文エントリ
72B0	STOP	STOP ON/OFF/STOP 処理
72C8	PEN	PEN 文エントリ
72CD	I/O イニシャライズ	N_{88}-BASIC 用 I/O をイニシャライズする。
7348	TERM サブルーチン	

アドレス	項　目　名	機　　　能
7367	TERM	TERM エントリ
779C	モード分岐	イニシャライズ時にターミナルモード，N-BASIC モードへ分岐するルーチン。
77DD	NEW	NEW エントリ
77E5	NEW ON	DE ← (NEW ON 値) & JMP
77F7	システム・イニシャライズ	リセットから飛んでくる。
7984 ∫ 7998		文字列 'How many files (0 ～ 15)', 0
79B2 ∫ 79FB	クレジット	文字列 ' Bytes free', 0 文字列 'NEC　N-88 BASIC......', 0
79FC ∫ 7A95	I/O 処理ルーチン アドレス	DISK 以外のデバイスに対する I/O 処理ルーチンのアドレステーブル 各デバイスにつき22バイト，11コの処理 LPT 1，COM 1，COM 2・3，CAS 1，CAS 2，SCRN，KYBDの順　cf. 4E7C，4E8C
7A96	OPEN　(DEVOPN)	FCB を OPEN 状態にする。
7AC0	GET/PUT の大きさ	DISK 以外のデバイスに対する GET/PUT の大きさの処理ルーチン
7B25	LPT FPOS	LPT FPOS 処理
7B39	LPT OPEN	LPT OPEN 処理
7B59	LPT CLOSE	LPT CLOSE 処理
7B76	LPT PUT	LPT PUT 処理
7BA5	LPT OUTPUT	LPT OUTPUT 処理
7BBE	LPT WIDTH	LPT WIDTH 処理
7BC2	COM 1 OPEN	COM 1 OPEN 処理
7C3B	COM 1 INPUT	COM 1 INPUT 処理
7C43	COM 1 OUTPUT	COM 1 OUTPUT 処理
7C57	COM 1 CLOSE	COM 1 CLOSE 処理
7C6E	COM 1 LOC	COM 1 LOC 処理
7C76	COM 1 LOF	COM 1 LOF 処理
7C7E	COM 1 EOF	COM 1 EOF 処理
7C8B	COM 1 BACK	COM 1　BACK READ DATA 処理
7C96	COM 1 WIDTH	COM 1 WIDTH 処理
7C9B	COM 1 1文字 INPUT	Acc ← 1 & CALL　Acc ←データ
7CCD		'Line buffer overflow' エラー出力

アドレス	項　目　名	機　　能
7D71	COM	COM エントリ
7DC3	カセット OPEN	カセット OPEN 処理
7E3A	カセット CLOSE	カセット CLOSE 処理
7E41	カセット GET/PUT	カセット GET/PUT 処理
7E7B	カセット BACK	カセット BACK READ DATA
7E82	カセット OUTPUT	カセット OUTPUT 処理
7E8A	カセット INPUT	カセット INPUT 処理
7EBA	ボーレイト	カセットボーレイト設置 (F009 ＝ FA　1200ボー，FB 600ボー)
7ED0	カセット READ ON	カセットリード用イニシャライズ（F009 ＝ボーレイト）
7F15	カセット READ OFF	カセットリード中止
7F1A	カセット WRITE OFF	カセットライト中止
7F30	MOTOR	MOTOR エントリ
7F35	MOTOR	MOTOR ON/OFF Acc ≠ 0 : MOTOR ON ＝ 0 : MOTOR OFF
7F4D	カセット WRITE ON	カセットライト用イニシャライズ（F009 ＝ボーレイト）
7F87	カセットリード	カセットから1文字入力する。データ→ Acc
7FD0	カセットライト	カセットへ1文字出力する。Acc ←データ & CALL

〔ROM 5　4thROM＃1〕アドレス6000～7FFF

アドレス	項目名	機能
600D ∫ 604A	ROM 5 ジャンプテーブル(2バイト×31組)	メイン ROM での ROM 5 コール CALL 4551 DB 〈処理番号〉 での，処理番号に対するジャンプテーブル（別表）
604B ∫ 605C	CIRCLE 文サブルーチンⅠ	スクリーンモードに合わせて縦、横の比を求める。
605D ∫ 625F	GET@ 文, PUT@ 文サブルーチン	GET@ 文、PUT@ 文のサブルーチンで，各種ワークの設定，および実際のパターンの読み出し、書き込みを行う。
6260 ∫ 6269	スクリーンモードチェックⅠ (Acc)	スクリーンモードによって、Acc をセットする。 （カラーモード……Acc ＝ 3 B／W モード……Acc ＝ 1）
626A ∫ 64BF	PAINT 文サブルーチンⅠ	PAINT 文のサブルーチンの集まり。（大きく分けて，7つのサブルーチンがある）
64C0 ∫ 6508	VIEW PORT 内チェック	オリジナル・スクリーン座標 (X, Y)（BC，DE レジスタ）が VIEW PORT 内にあるかどうかをチェックする。 （内……CY ＝ 1 外……CY ＝ 0）
6509 ∫ 6517	CIRCLE 文サブルーチンⅡ	FAC ＝ FAC ＊ FRX の計算（FAC には，円の半径が入っている）
6518 ∫ 657B	CADDR, CMASK の計算	オリジナルスクリーン座標 (X, Y) からグラフィック画面の物理アドレス (CADDR)，データパターン (CMASK) を求める。
657C ∫ 65F1	LINE 文サブルーチンⅠ	LINE 文で，点を上下左右に移動させたときの，CMASK，CADDR を求める。（一部，PAINT 文でも使用）
65F2 ∫ 661A	汎用サブルーチン　Ⅰ	
661B ∫ 6646	POINT (Sx, Sy) 関数サブルーチン	POINT (Sx, Sy) 関数で，指定した点のパレット番号を Acc に入れてもどる。
6647 ∫ 669F	LINE BF 文サブルーチン	LINE...BF 文で指定した領域を塗りつぶすサブルーチン。

アドレス	項 目 名	機 能
66A0 ～ 66D6	CURAT セット	パレット番号によって CURAT 1 ～ 3 に 0 または，FF を書きこむ。
66D7 ～ 66FF	グラフィックデータ書き込み	指定したカラー (CURAT 1 ～ 3) で，(CMASK) のデータを書く。
6700 ～ 6877	グラフィックスクリーンの初期化	リセットまたは、ホットスタート時に呼び出される。 グラフィックスクリーンをクリアし，各種ワークエリア等を初期化する。
6878 ～ 698E	COLOR 文	COLOR 文の処理を行う。 COLOR………6882 COLOR＝……68EC COLOR@……6927
698F ～ 6A93	SCREEN 文	SCREEN 文の処理を行う。
6A94 ～ 6AC5	CLS 2 文	CLS 2 文の処理を行う。
6AC6 ～ 6C2A	VIEW 文	VIEW 文の処理を行う。
6C2B ～ 6C54	WINDOW 文 サブルーチン	WINDOW 文の座標パラメータを得る。
6C55 ～ 6CA7	WINDOW 文	WINDOW 文の処理を行う。
6CA8 ～ 6CD4	VIEW 関数	VIEW 関数の処理を行う。
6CD5 ～ 6D09	WINDOW 関数	WINDOW 関数の処理を行う。
6D0A ～ 6D28	LP の初期化	LPの値を左上の座標にセットする。 (SCREEN 文，VIEW 文，WINDOW 文の実行後)

アドレス	項目名	機能
6D29 〜 6DBF	MAP 関数	MAP 関数の処理を行う。 W → S……6D8B S → W……6DAB
6DC0 〜 6E24	POINT(〈機能〉)関数	POINT 関数の処理を行う。
6E25 〜 6E2A	POINT 文	POINT 文の処理を行う。
6E2B 〜 6EF7	座標パラメータ （ワールド座標系）	テキストから座標パラメータ（ワールド座標系）を得る。 （WINDOW 文が実行されていなければ，スクリーン座標系となる。）
6EF8 〜 6F32	スクリーン座標 →ワールド座標	スクリーン座標からワールド座標への変換を行う。
6F33 〜 D052	LINE 文サブルーチン Ⅱ	LINE 文で，線がオリジナルスクリーン上に書けるかどうかを調べる。
7053 〜 723C	画面コピー	画面コピー（COPY 文，COPY キー）の処理を行う。
723D 〜 72CF	漢字データ読み出し	PUT@ KANJI 文のサブルーチンで漢字データをワークエリア上に読み出す。
72D0 〜 72E0	ON ×× GOSUB サブルーチン Ⅰ	Line ＃（DE レジスタ）を Acc 番目のジャンプテーブルにセットする。
72EC 〜 72E1	Read Light Pen	ライトペン入力信号が来たときの行，列座標を読み出す。
72ED 〜 7323	PEN 関数	PEN 関数の処理を行う。
7324 〜 73DE	ON ×× GOSUB サブルーチン Ⅱ	ON ×× GOSUB の前処理を行う。××によって，BC レジスタをセットして，メイン ROM へ戻る。

アドレス	項目名	機能
73DF 〜 74ED	KEY 文	KEY 文の処理を行う。 KEY LIST..............73DF KEY 定義..................7443 KEY OFF................7484 KEY ON...................748D KEY STOP.............749C KEY (　) ON, OFF, STOP......74A1
74EE 〜 7529	DATE$ 関数	DATE$ 関数の処理を行う。
752A 〜 754E	TIME$ 関数	TIME$ 関数の処理を行う。
754F 〜 75A0	ドライブ対応表作成	ドライブ番号→ドライブタイプの対応表を作成する。
75A1 〜 75DC	2W モードセット	PC-8031-2W がつながっていれば，2W モードにする。
75DD 〜 7673	RENUM 文	RENUM 文の処理を行う。
7674 〜 77DD	PAINT 文	PAINT 文の処理を行う。
77DE 〜 77E3	CIRCLE 文 サブルーチン　Ⅲ	DE ＝ − DE を求める。
77E4 〜 79D5	PAINT 文 サブルーチン　Ⅱ	PAINT 文のサブルーチン，主にタイルストリング関係の処理を行う。
79D6 〜 7C32	CIRCLE 文	CIRCLE 文の処理を行う。
7C33 〜 7D97	GET@，PUT@ 文	• GET@，PUT@文の処理を行う。 (GET@......(F071)＝ 0) (PUT@......(F071)＝ 1)

アドレス	項　目　名	機　　能
7D98 ∫ 7DFA	座標パラメータ （スクリーン座標系）	テキストから座標パラメータ（スクリーン座標系）を得る。
7DFB ∫ 7E18	PRESET，PSET 文	PRESET 文の処理を行う。...7DFB PSET 文　　〃　　......7E00
7E19 ∫ 7E33	POINT (Sx,Sx) 関数	POINT (Sx, Sy) 関数の処理を行う。
7E34 ∫ 7E54	パレット番号を得る	テキストからパラメータ（パレット番号）を得る。
7E55 ∫ 7E8A	汎用サブルーチン　II	HL ＝　GXPOS-BC......7E55 HL ＝　GYPOS-DE......7E67 DE ⟷ GYPOS.........　7E72 BC ⟷ GXPOS......　　757F
758B ∫ 7FBC	LINE 文	LINE 文の処理を行う。 LINE BF...... 73B7 LINE のみ......7EF2 LINE B......... 7F0B
7EBD ∫ 7FC4	汎用サブルーチン　III	DE ＝ DE ¥2
7FC5 ∫ 7FD4	LINE 文サブルーチン III	ラインスタイルを見て，点を打つ。
7FD5 ∫ 7FEC	汎用サブルーチン　IV	テキストから1バイトのパラメータを得る。
7FED ∫ 7FFA	HS ディスプレイ チェック	専用高解像度ディスプレイモードかどうかを調べる。

〔ROM 5〕処理番号と処理開始アドレス

処理番号	アドレス	内　　　容
0	7DFB	PRESET文
1	7E00	PSET文
2	6D29	MAP関数
3	7C33	GET@，PUT@文
4	7053	COPY文
5	7064	COPYキー
6	7E8B	LINE文
7	6E25	POINT文
8	6700	グラフィック画面初期化
9	6AC6	VIEW文
10	7E19	POINT（Sx，Sy）関数
11	6DC0	POINT関数
12	6878	COLOR文
13	EEBC	ROLL文
14	79D6	CIRCLE文
15	6CD6	WINDOW関数
16	6C55	WINDOW文
17	7674	PAINT文
18	698F	SCREEN文
19	6CA8	VIEW関数
20	6A94	CLS2
21	6D0A	LPの初期化
22	72D0	ON××GOSUBサブルーチンⅠ
23	72ED	PEN関数
24	7324	ON××GOSUBサブルーチンⅡ
25	742E	KEY文
26	752A	TIME＄関数
27	74EE	DATE＄関数
28	75A1	PC－8031－2Wモードセット
29	754F	ドライブ対応表作成
30	75DD	RENUM文

付録3　N_{88}-DISK-BASIC インタプリタ解析

[Apr. 24, 1982] Version.

アドレス	項　目　名	機　　　能
8400	モニタ・ジャンプテーブル初期化	モニタコマンド・ジャンプテーブル初期化
8427	モニタ　^Dコマンド	フロッピィディスクの内容を表示する。
84F0	モニタ　^Rコマンド	フロッピィディスクからデータをロードする。
84F1	モニタ　^Wコマンド	フロッピィディスクにデータをセーブする。
8588 ∫ 8621	モニタ　サブルーチン（ディスク関係）	^D, ^R, ^W, コマンド用のサブルーチンの集まり。
8622	モニタ　ワークエリア	^D, ^R, ^Wコマンドで，ディスクのサーフェイスを表す。
8623	モニタ　HELPコマンド	HELPコマンドの拡張部分（ディスク用コマンドの説明を出力する）。
866A ∫ 8769	データ	HELPコマンド用のメッセージ
876A	モニタ　Mコマンド	Mコマンドの完全バージョン
8790	モニタ　サブルーチン	ディスク情報をワークエリアにコピーする。
8796	モニタ　サブルーチン	ドライブポインタをセットする。 ———以上モニタ関係のルーチン———
87D6	ROLL UP　後処理	キーバッファをクリアして，EDITへ戻る。
87E6	ROLL DOWN　後処理	キーバッファをクリアして，EDITへ戻る。
87F4	CHAIN文サブルーチン	ラベル スキップ
8803	ROLL文	ROLL文の処理を行う。
88E9	SEARCH関数	SEARCH関数の処理を行う。
89B3	ATN関数（単精度）	ATN（FAC）→FAC

アドレス	項目名	機能
89D7 〜 89FB	データ	ATN関数用のデータ
89FC	ディスク情報セット	ディスク情報をワークエリア(ECA5〜)にセットする。
8A18 〜 8A3B	データ	ディスク情報データ
8A3C 〜 8DE4	データ	フルセンテンス エラーメッセージ
8DE5	テキスト入力時の処理	MAINループでのテキスト入力時に，CHAIN文であればストリングエリアをクリアしないようにするためのルーチン。
8E4B	CHAIN クリア	エラーがおこったとき，CHAIN文フラグをクリアする。
8E55	デバイス デフォルトセット	ファイルディスクリプタのデバイス番号のデフォルト値を0にする。
8E58	イニシャライズ	ドライブテーブルを初期化して，IDセクタを実行する。
8E9A	GET デバイス♯	ファイルディスクリプタのデバイス♯をAccへ入れる。
8EC3	DSKF 関数	DSKF関数の処理を行う。
8F41	式の評価	式の評価ルーチンを倍精度関数用に拡張
8F7C	倍精度分岐	倍精度関数ルーチンへのふり分けを行う。
8F8E 〜 8F9D	倍精度関数ジャンプテーブル	SQR，RND，SIN，LOG，EXP，COS，TAN，ATN
8FA4	DISK関数評価	DSKI$，INPUT$，ATTR$ の分岐 また、USR関数実行時に，プロテクトのチェックを行う。
8FBD	ファイル関数評価	EOF、FPOS，LOC，LOFの分岐
8FDD	フルセンテンスエラーメッセージセット	フルセンテンスのエラーメッセージのポインタをセットする。
9001	KYBD：OPEN	OPEN "KYBD" の処理
900C	KYBD：INPUT♯	〃 INPUT♯ の処理
9018	KYBD：GET♯	〃 GET♯ の処理
9032	KYBD：READ BACK	〃 READ BACK の処理
903C	KYBD：LOC	〃 LOC関数の処理
9043	SCRN：OPEN	OPEN "SCRN" の処理
9051	SCRN：OUTPUT♯	〃 OUTPUT♯の処理
905F	SCRN：PUT♯	〃 PUT♯の処理

アドレス	項目名	機能
90C4	WEND サーチ	WENDをサーチする。
90C9	ワークエリア	CHAIN文で使用する。OPTION BASEのフラグと値
90CC	ROLL UP	EDITモードでのROLL UPキーの処理
9118	ROLL DOWN	EDITモードでのROLL DOWNキーの処理
91A5	EDIT 行番号 GET	EDITモードのための行番号を画面から読みとる。
9247 ∫ 9325	倍精度関数サブルーチン	倍精度関数計算のための汎用サブルーチン集
9326 ∫ 948A	データ	倍精度関数計算のための数値データ
948B	ATN (倍精度)	ATN (FAC) →FAC
94E7	COS (倍精度)	COS (FAC) →FAC
94F0	EXP (倍精度)	EXP (FAC) →FAC
956F	LOG (倍精度)	LOG (FAC) →FAC
9615	SIN (倍精度)	SIN (FAC) →FAC
9650	SQR (倍精度)	SQR (FAC) →FAC
9691	TAN (倍精度)	TAN (FAC) →FAC
9717	ファイル削除	FATからファイルを削除する。
973B	MOUNT メインルーチン	ディスクをマウントする。
979F ∫ 97C2	データ	DB 'Copies of allocation bad on drive'
97FD	REMOVE サブルーチン	REMOVEルーチンのサブルーチン，FATの更新を行う。
97EA	REMOVE	リムーブ処理を行う。(OPEN，CLOSE，KILL) IN：デバイス#
9821	MOUNT	デバイスがディスクだったらマウント処理を行う。
982C	ファイル#チェック	PUT#，GET#で，ファイル#が現在のファイルのレコード数より大きいかどうかを調べる。
9869	LOC関数サブルーチン	クラスタ数を数える。
9886	FAT TOPセット	DEレジスタにFATの先頭番地をセットする。
98DA	クラスタ→セクタ	クラスタ数からセクタ数を求める。 IN：C ＝クラスタ数 OUT：DE＝セクタ数
98EA	ディレクトリサーチ	ディレクトリからファイル名を探す。(なければ2F＝)

アドレス	項 目 名	機 能
9953	ディスク情報セット	ディスク情報，ドライブポインタをセットする。 IN：Acc＝ドライブ♯
9997	IDセクタ計算	BCレジスタにIDトラック，セクタをセットする。
99A1	ディレクトリトラック	Bレジスタにディレクトリトラックをセットする。
99B6	NAME文	NAME文の処理を行う。
9A01	ディスクファイル OPEN	ディスクファイルをオープンする。
9B21	ディスクファイル CLOSE	ディスクファイルをクローズする。
9BAD	KILL文	KILL文の処理を行う。
9BC5	ディスクSAVE	ディスクへのセーブルーチン
9C26	BSAVEサブルーチン	
9C35	BLOADサブルーチン	
9C51	ディスクLOAD	ディスクからのロードルーチン
9CE7	OPENチェック	ファイルがすでにオープンされているか調べる。
9DAD	SET文	SET文エントリ
9E1D	ATTR$関数	ATTR$関数エントリ
9E5E	SET文，ATTR$	SET文，ATTR$関数のパラメータを処理する。
9F2A	LFILES文	LFILES文エントリ
9F2F	FILES文	FILES文エントリ
9FFD	PUT♯／GET♯	ランダムファイルの読み書きを行う。（ディスク）
A104	INPUT$	INPUT$文の処理（ディスク）
A185	DSKI$関数	DSKI$関数の処理を行う。
A1C4	DSKO$文	DSKO$文の処理を行う。
A1F3	DSKI$，DSKO$サブ	DSKI$，DSKO$のパラメータをレジスタにセットする。
A2F7	FAT READ	FATをメモリ上に読み出す。
A327	ディレクトリREAD	ディレクトリをメモリ上に読み出す。
A340	ディレクトリWRITE	ディレクトリを書き込む。
A4E0	カレントセクタ READ	現在のファイルの1セクタを読み出す。
A4FF	トラック・セクタ計算	カレントクラスタ・セクタからトラック・セクタを計算する。
A53B	DISK READ／ WRITE	ディスクのREAD／WRITEルーチン
A61A	DSKF関数	ディスクのフリークラスタを求める。
A640	LOC関数	
A669	LOF関数	
A685	EOF関数	

アドレス	項　目　名	機　　　能
A6C9	FPOS関数	
A6F1	IDセクタREAD	IDセクタをキーバッファに読み込む。
A710	エラー69	Bad allocation table
A713	エラー70	Bad drive number
A716	エラー71	Bad track sector
A719	エラー67	Disk already mounted
A71C	エラー63	Disk not mounted
A71F	エラー72	Deleted record
A722	エラー65	File already exists
A725	エラー68	Disk full
A728	エラー61	File write protected
A72B	エラー64	Disk I/O error
A72E	エラー62	Disk offline
A731	エラー73	Rename across disks.
A73D	BSAVE文	BSAVE文の処理を行う。
A7A8	BLOAD文	BLOAD文の処理を行う。
A810	BLOAD,BSAVE サブ	
A875	WHILE文	WHILE文の処理を行う。
A89F	WEND文	WEND文の処理を行う。
A916	CALL文	CALL文の処理を行う。
A98C	CHAIN文	CHAIN文の処理を行う。
AC10	CHAIN文ジャンプ	CHAIN文で指定された行番号へとぶ。
AC72	COMMON文	COMMON文の処理を行う。
AC75	WRITE文	WRITE文の処理を行う。
ACC1	SAVE 暗号化	Pオプションセーブのときにプログラムを暗号化する。
AD05	LOAD 平文化	Pオプションセーブをしたプログラムをロードしたあともとに戻す。
AD48 ≀ ADA1	プロテクトチェック	プロテクト（Pオプションセーブしたプログラムがロードされている）の状態であれば，以下のコマンドをエラーとしてしまう。 LIST, SAVE, MON, USR, CALL, PEEK, POKE
	以　上 DISK-BASIC本体	
AF00 ≀ B291	イニシャライズルーチン	最初にイニシャライズを行ったあとは不要になる。

付録4　N_{88}-BASIC　モニタルーチン解析

アドレス	項　目　名	機　　能
6000	データ	'DB'
6002	モニタ スタート	
6047	RST　38H	ブレークポイントからとんでくる。
6088	モニタ メインループ	
60F0 ∫ 6105	コマンド テーブル	モニタで使用できるコマンド(22個)。
6106 ∫ 6131	コマンドアドレステーブル	各コマンドに対する処理アドレスのテーブル。 (2バイト×22組)
6135	*Sコマンド	メモリ内容を書き換える。
61A3	*Dコマンド	メモリ内容を出力する。
6254	*Xコマンド	CPUレジスタに関するコマンド。 CPUレジスタを変更する。………6260 CPUレジスタを全て出力する。…62DA
63D8	*Fコマンド	メモリ内容を定数で埋める。
6406	*Gコマンド	ユーザー・プログラムを実行する。
6479	*Iコマンド	入力ポートの値を読み込む。
6491	*Oコマンド	出力ポートへデータを出力する。
64A6	*Mコマンド	メモリの内容を転送する。
64D8	*Eコマンド	スクリーンエディタでメモリ内容を変更する。 サブコマンドテーブル　6541～654B サブコマンドアドレステーブル　654C～6561 CTRL-f　65BD CTRL-b　65C4 up arrow　65CB down arrow　65EC right arrow　661B left arrow　662F

アドレス	項目名	機能
		ROLL UP 6649
		ROLL DOWN 669C
		editから抜ける 66EE
6754	*Wコマンド	メモリ内容をカセットテープにセーブする。
6806	*V, Rコマンド	Vコマンド…6806 カセットテープの内容をベリファイする。 Rコマンド…6807 カセットテープからロードする。
695E	*Bコマンド	8進，16進の切り換えを行う。
6976	*Lコマンド	機械語をディスアセンブルする。
6D02	*Aコマンド	入力行をアセンブルする。
6E5F	*^Bコマンド	BASICへ戻る。
6E7A	*Pコマンド	プリンタスイッチを切り換える。
6E8A	プリンタ出力	画面に出力した行をプリンタに出力する。
6EEE	GETパラメータ	コマンド行からパラメータを得る。(HLレジスタ)
6F80	数値出力Ⅰ	1バイトの数値を出力する。(Acc)
6F9A	数値出力Ⅱ	2バイトの数値を出力する。(HLレジスタ)
6FA7	文字列出力	文字列を出力バッファに書き込む。
6FCF	BIN→ASCII	BINコードをASCIIコードに変換する。
6FD7	HL，DE比較	HLレジスタとDEレジスタとを比較する。
6FDD	1文字入力	キーボードから1文字入力する。
7025	小文字→大文字	小文字から大文字に変換する。
702E	文字列出力Ⅰ	(HL) から (HL) ＝0の前までを出力する。
7037	CR，LF出力	CR，LFコードを出力する。
7044	スペース出力	スペース文字を出力する。
704C	文字列出力Ⅱ	(HL) から (HL) のビット7が1になる文字までを出力する。
7058 ∫ 7128	メッセージ	無意味なメッセージ⁉
7129	1文字出力Ⅰ	Accの値を画面に出力する。
7159	キー入力	キーボードからの1文字入力を行う。
7162	キーセンス	キーボードが押されているか調べる。 (押されていれば CY＝0)
716E	VRAMアドレス	カーソル位置からVRAMアドレスを得る。
717B	カーソルON	カーソル表示をONにする。
7183	カーソルOFF	カーソル表示をOFFにする。
718B	カセットWRITE ON	カセットインターフェイスへの書き込みを開始する。
7194	カセット出力	カセットインターフェイスへデータを出力する。(Acc)

アドレス	項　目　名	機　　　能
719D	カセットWRITE OFF	カセットインターフェイスへの書き込みを終了する。
71A6	カセットREAD ON	カセットインターフェイスからの読み出しを開始する。
71AF	カセットREAD OFF	カセットインターフェイスからの読み出しを終了する。
71B8	カセット入力	カセットインターフェイスからデータを入力する。(Acc)
71C3	LINE INPUT	1行入力を行う。
71CC	CRTC セット	CRTC（μPD3301）の設定を行う。
71D5	BUSYチェック	プリンタからのBUSY信号をチェックする。
71DA	1文字出力Ⅱ	プリンタへ1文字出力する。
71F4 ~ 726C	データ	RAM上（F155～F1CD）に置かれるサブルーチンのデータ。
7270	メインROM　LDIR	メインROMをセレクトし，ブロック転送（LDIR）を行う。
7276	メインROM　LDDR	メインROMをセレクトし，ブロック転送（LDDR）を行う。
727C	*TMコマンド	メモリのテストを行う。
746F	HELPコマンド	コマンド一覧表を出力する。
77E8	Edit　HELP	スクリーンエディタ時のサブコマンド一覧表を出力する。
7973	フラグHELP	フラグをセットする時に，フラグの説明を出力する。
F155	N_{88}-BASIC ROM コール	N_{88}-BASIC ROMをコールする。 CALL　F155 DW　ADRS
F16C	ROMセレクト	N-BASIC ROM，N_{88}-BASIC ROMのセレクトを行う。
F184	ブロック転送	ブロック転送ルーチン LDIR…F184 LDDR…F187
F18A	メモリアクセス	メインROM，RAMをセレクトし，メモリをアクセスする。
F1BC	エラールーチン	モニタROMをセレクトし，エラールーチンへ。
F1C2	GOコマンド用テーブル	GOコマンドでメインROMをセレクトしてジャンプする。
F1C8	ブレークポイント	RST38Hでここへとんでくる。モニタROMをセレクトし，6047へ。

付録5　ワークエリア一覧表

アドレス	ラ ベ ル	機 能・用 途
E600～0D	FDIVC	単精度除減算用サブルーチン
E60E		RND関数　発生カウンタ
E60F	RNDCNT	RND関数　ROMデータ　カウンタ
E610		RND関数　RAMデータ　カウンタ
E611～30	RNDTAB	RND関数　RAMデータ（単精度）　4バイト×8組
E631～34	RNDX	RND関数の値
E635～48	USRTAB	USR関数　アドレステーブル　2バイト×10組 （初期値　B06：Illegal function call）
E649	ERRFLG	ERRの値　（ERL＝EB09）
E64A		未使用
E64B	LPTPOS	LPOSの値
E64C	PRTFLG	プリンタフラグ （出力先の指定，0：CRT，0以外：LPT）
E64D	NLPPOS	LPRINT‘，’改行桁数
E64E	LPTSIZ	WIDTH　LPRINTの値
E64F	LINLEN	PRINT‘，’改行桁数
E650	CLMLST	PRINT‘，’改行桁数
E651		未使用
E652	CNTOFL	CTRL＋Oフラグ （画面出力をさせないためのフラグ）
E653	FLBMEM	BLOAD／BSAVE用フラグ （0：アドレス指定のみ，1：指定なし，FF：アドレス・長さあり）
E654，55	TOPMEM	メモリ上限値，CLEAR文の第2パラメータ
E656，57	CURLIN	実行中の行番号
E658，59	TXTTAB	テキスト開始アドレス
E65A，5B	OVERRI	OV，／0エラーメッセージアドレス

アドレス	ラベル	機能・用途
E65C～68	NECID	ターミナルID
E669～6B	MONRMI	モニタ　ブレークポイント用ジャンプテーブル（RST38Hで使用）
E66C～6E		モニタ　イニシャライズ拡張用フック
E66F～71		モニタ　拡張Mコマンド用フック
E672～74		モニタ　^Dコマンド・ジャンプテーブル
E675～77		モニタ　^Wコマンド・ジャンプテーブル
E678～7A		モニタ　^Rコマンド・ジャンプテーブル
E67B～7D		モニタ　Eコマンド拡張用フック
E67E～80		モニタ　拡張HELPコマンド用フック
E681～83		モニタ　汎用フック
E684～9B		未使用
E69C, 9D	NUMCOM	RST 10用 work
E69E	DEVTYP	前にアクセスしたドライブタイプ （8インチ＝0，5インチSS＝2，5インチDS＝3）
E69F	TRMCMD	TERM中を示すフラグ
E6A0	TRMREM	TERM　リモートプロトコル用フラグ （FF：ESC>&ノーマル，00：ESC.，01：ESC<）
E6A1	TRMCHR	TERM　halfキー入力　1メッセージ最初の文字
E6A2	TRMLPF	TERM　LPT イネーブルフラグ　（f・8）
E6A3	IEEINF	READで0だったらOut of Data
E6A4	CASEOF	カセット　ロード中…FF，OPEN…1， 1Aを入力のとき…0
E6A5	CL.FLG	TERM　LF/COPYフラグ （LF…1，COPY…2）
E6A6	HIRESL	ハイレゾモードフラグ （640×200…0，640×400…1）
E6A7	CURFG	カーソルフラグ
E6A8	CURFG2	カーソル　ON／OFF　コマンド
E6A9	CASPRT	カセット使用中フラグ （READ…1，WRITE…FF）
E6AA		未使用
E6AB, AC	DOT	‘.’行番号
E6AD	INSFLG	INSモードフラグ
E6AE, AF	PENTBL	ライトペン補正値
E6B0	SCRLL	テキストウィンドウ上限（実際のウィンドウ）
E6B1	LINEND	テキストウィンドウ下限（　〃　）
E6B2	SCRLL1	テキストウィンドウ上限CONSOLE A, Bで→A＋1の値
E6B3	SCRLL2	テキストウィンドウ下限CONSOLE A, Bで→A+B＋1の値

アドレス	ラ ベ ル	機能・用途
E6B4	NULATR	ヌルアトリビュート
E6B5	NULCHR	ヌルキャラクタ・コード
E6B6	F.LTRL	コントロールコードをCRTに出力するフラグ
E6B7		未使用
E6B8	CNSDFG	ファンクションキー表示フラグ
E6B9	CMODE	テキストモード（カラー…FF，B／W…0）
E6BA	F.COPY	コピーモード （ノーマル…FF，テキスト…FD，グラフィック…FA）
E6BB，BC	SCUDOT	EDIT 一番上の行番号
E6BD，BE	SCDDOT	EDIT 一番下の行番号
E6BF	ERRCST	8251エラー発生フラグ
E6C0	S.SYC0	PORT 30Hに出力したデータ
E6C1	S.SYC1	PORT 40Hに出力したデータ
E6C2	S.CRTC	PORT 31Hに出力したデータ
E6C3	S.ILVL	PORT E4Hに出力したデータ
E6C4，C5	VRAMAD	テキストVRAMアドレスTOP（初期値：F3C8）
E6C6	WAITFG	(LINE) INPUT WAITフラグ
E6C7	TIMRFG	ON TIME$ カウンティングフラグ
E6C8		未使用
E6C9	CBOEFL	Communication Buffer Overflow フラグ
E6CA	INTFLG	イベントキーフラグ（^O，^S，^C）
E6CB，CC	QUEUES	キューテーブルアドレス
E6CD	F.KSUP	キー入力サプレスフラグ
E6CE	F.KYST	ファンクションキー ストア中フラグ
E6CF	KEYPRT	キー入力 リピートカウンタ
E6D0～DB	KEYBIT	キー入力ポートデータのCPL キーステータステーブル1
E6DC～E7	KEYVLD	キー入力押している所＝1 キーステータステーブル2
E6E8	CASACT	カセット使用中フラグ（WRITE…FF，READ…1）
E6E9，EA	LPDTCT	COPY文 ドット対応グラフィック出力値カウンタ
E6EB	LPTBSY	LPT OPENフラグ
E6EC	COMSIZ	COM1のWIDTH
E6ED	RSCOMD	RS-232C使用中フラグ （8251用コマンド）
E6EE	TRPNUM	ON割込みの全部の個数
E6EF，F0	TRPTBA	ON割込みテーブルアドレス

アドレス	ラ ベ ル	機 能・用 途
E6F1	ONGSBF	ON割込みフラグ（カウンタ）
E6F2～E7E1	STRTAB	ファンクションキーデータ(16バイト×15組）
E7E2～E4	MONUPD	LINE文 サブルーチン ジャンプテーブル1
E7E5～E7	MAXUPD	LINE文 サブルーチン ジャンプテーブル2
E7E8，E9	MAXMEM	CLEARできる上限（E5FF）
E7EA～E825	INT0	割込み処理ルーチン E7EA…RS-232C エントリ E808…VRTC エントリ E80E…CLOCK エントリ E814…USER エントリ E81A…DISK1 エントリ E820…DISK2 エントリ
E826～4F	MON	モニタルーチン
E850	NAMCNT	変数名 3文字目以降の長さ
E851～76	NAMBUF	変数名バッファ（3文字目以降）
E877，78	NAMTMP	配列変数テキストポインタ退避
E879		（未使用）‘；’が入っている
E87A～E9B7	KBUF	中間言語バッファ （最後の数バイトは使用されない）
E9B8		（未使用）
E9B9～EABA	BUF	行入力バッファ
EABC	DIMFLG	変数認識ルーチン中でDIM文から呼ばれたことを示す。
EABD	VALTYP	FACのタイプ
EABE	DORES	中間言語⟷リストルーチン，DATA文， “…”の中などを示す
EABF	DONUM	中間言語←リスト，数字→行番号 フラグ
EAC0，C1	CONTXT	RST 10H テキストポインタ
EAC2	CONSAV	RST 10H 読んだ文字
EAC3	CONTYP	RST 10H FACのタイプ
EAC4～CB	CONLO	RST 10H FAC
EACC，CD	MEMSIZ	文字列エリアの始まり
EACE，CF	TEMPPT	ストリングスタックポインタ
EAD0～ED	TEMPST	ストリングスタック（3バイト×10組）
EAEE～F0	DSCTMP	ストリングディスクリプタ
EAF1，F2	FRETOP	フリーエリアの終り
EAF3，F4	TEMP3	Work
EAF5，F6	TEMP8	Work（ガベージ コレクション）

アドレス	ラ ベ ル	機 能・用 途
EAF7, F8	ENDFOR	FOR文テキストポインタ退避
EAF9, FA	DATLIN	DATA文エラー行番号
EAFB	SUBFLG	認識すべき変数のタイプ
EAFC	USFLG	READ／INPUTフラグ，USING ルーチンフラグ
EAFD, FE	TEMP	Work
EAFF	PTRFLG	行番号→行アドレス フラグ
EB00	AUTFLG	AUTOモードフラグ
EB01, 02	AUTLIN	AUTOモードで次に発生させる行番号
EB03, 04	AUTINC	AUTOモード増分
EB05, 06	SAVTXT	1文実行前の(ステートメント)アドレス
EB07, 08	SAVSTK	1文実行前のスタック
EB09, 0A	ERRLIN	ERLの値
EB0B, 0C	ERRTXT	エラーを起したアドレス
EB0D, 0E	ONELIN	ON ERROR GOTO のとび先アドレス
EB0F	ONEFLG	エラートラップルーチン中フラグ (FF＝トラップルーチン)
EB10, 11	TEMP 2	Work
EB12, 13	OLDLIN	実行停止時(END，STOP)のときの行番号
EB14, 15	OLDTXT	CONT実行再開アドレス
EB16, 17	LBLTAB	ラベル領域開始アドレス
EB18, 19	TXTEND	テキストエンド＋1
EB1A	LBLFLG	ラベル登録済みフラグ
EB1B, 1C	VARTAB	単純変数領域開始アドレス
EB1D, 1E	ARYTAB	配列変数領域開始アドレス
EB1F, 20	STREND	フリーエリアの始まり
EB21, 22	DATPTR	READ文データポインタ
EB23～3C	DEFTBL	変数の暗黙型
EB3D, 3E	PRMSTK	FN引数スタックポインタ
EB3F, 40	PRMLEN	FN引数テーブル1の長さ
EB41～A4	PARM 1	FN引数テーブル1
EBA5, A6	PRMRV	PRMSTKを指す
EBA7, A8	PRMLN 2	FN引数テーブル2の長さ
EBA9～EC0C	PARM 2	FN引数テーブル2
EC0D	PRMFLG	FN引数テーブル1サーチ中フラグ
EC0E, 0F	ARYTA 2	単純変数／FN引数サーチ終りアドレス＋1
EC10	NOFUN	FNテーブルサーチフラグ (FN中フラグ)
EC11, 12	TEMP 9	Work
EC13	FUNACT	FNネスティングレベル

アドレス	ラ ベ ル	機 能・用 途
EC15	INPPAS	INPUT文でデータの数をかぞえるために空読みするフラグ
EC16, 17	NXTTXT	NEXT文アドレス
EC18	NXTFLG	NEXT文フラグ （0：FOR文からきたことを示す）
EC19～1C	FLALSV	FOR文初期値
EC1D, 1E	NXTLIN	NEXT文行番号
EC1F	OPTVAL	OPTION BASE文の値
EC20	OPTFLG	OPTION BASE文実行済フラグ
EC21	TEMPA	中間言語→リスト のスペースコントロール／CALL文 Work
EC22		INPUT文での'？ 'を出力するかどうかのフラグ／CALL文 Work
EC23, 24	SAVFRE	CHAIN文 FRETOPの退避
EC25, 26		未使用
EC27	TPROFL	SAVE 暗号化フラグ
EC28	TPROFI	LOAD 平文化フラグ
EC29	PROFLG	プログラム プロテクトフラグ
EC2A	MRGFLG	CHAIN文 MERGEフラグ
EC2B	MDFLG	CHAIN文 DELETEフラグ
EC2C, 2D	CMEFLG	CHAIN文 DELETEアドレスEND
EC2E, 2F	CMSPTR	CHAIN文 DELETEアドレスTOP
EC30	CHNFLG	CHAIN文 フラグ
EC31, 32	CHNLIN	CHAIN文 実行行番号
EC33～3A	SWPTMP	SWAP用FRG（Flowtingpoint Register）
EC3B	TRCFLG	TRONフラグ
EC3C～44	FAC	FAC（浮動小数点アキュームレータ）
EC45		FAC 演算用フォーマットでの符号
EC46	FLGOVC	OV，／0エラーフラグ（演算）
EC47	OVCSTR	OVフラグ （文字列→数値）
EC48	FANSII	絶対値＜1のとき非指数形式にするフラグ
EC49～51	ARG	倍精度用FRG
EC52～69	FBUFFR	数値を文字列にするときのバッファ
EC6A～6C		未使用
EC6D～74		倍精度除算用Work
EC75～79		未使用
EC7A～7C		単精度乗算用Work
EC7D	MAXDRV	接続されているドライブ数
EC7E	MAXFIL	オープンできるファイル数

アドレス	ラベル	機能・用途
EC7F, 80	FILTAB	ファイルバッファ アドレステーブルのアドレス
EC81, 82	DRVPTR	ドライブポインタのアドレス
EC83, 84	NULBUF	ヌルバッファアドレス
EC85	CURDRV	カレント ドライブ番号
EC86, 87	DRVPTR	カレントドライブテーブルのアドレス
EC88, 89	FILPTR	カレント ファイルバッファのアドレス
EC8A, 8B	FREPLC	ディレクトリサーチ時のファイル名ポインタ
EC8C	LSTFRE	FREPLCのセクタ中のファイル番号
EC8D		FREPLCのセクタ番号
EC8E	FILMOD	ファイルモード／LOAD ,Rのフラグ
EC8F～97	FILNAM	ファイル名1
EC98～A0	FILNM2	ファイル名2
ECA1	LSTTRK	DSKO$, DSKI$ トラック番号
ECA2	LSTSCT	DSKO$, DSKI$ セクタ番号
ECA3	NLONLY	bit7：すべてのファイルをcloseしない。 bit0：ヌルバッファはcloseしない。
ECA4	SAVFLG	SAVEフラグ
ECA5	MAXTRK	最大トラック番号
ECA6	NUMSEC	1トラックあたりのセクタ数
ECA7	TWOSUR	片面＝0, 両面＝1
ECA8	CLSTRK	1トラックあたりのクラスタ数
ECA9	NUMCLS	ボリュームあたりのクラスタ数
ECAA	DIRTRK	ディレクトリトラック番号
ECAB	CLSIZ	1クラスタあたりのセクタ数
ECAC	FATONE	FATの開始セクタ番号
ECAD	FATLST	FATの終了セクタ番号
ECAE	FATNUM	FATの数
ECAF	DSKINF	ディスク属性の入っているセクタ番号
ECB0	MODCNT	未使用
ECB1, B2	SAVEND	BSAVE ENDアドレス
ECB3		未使用
ECB4	ERRCNT	DISK R／Wエラー回数 （DISK-BASICでは使われない）
ECB5	ERRCN1	DISK R／Wエラー回数 （4回になるとDISK I/Oエラー）
ECB6	RAWFLG	リードアフターライトフラグ
ECB7	EBCFLG	EBCDICコードフラグ
ECB8	SAVEBC	EBCFLG退避
ECB9, BA		未使用

アドレス	ラベル	機能・用途
ECBB～EE0D		フックアドレステーブル　（別表）
EE0E～70		I/O処理ルーチンジャンプテーブル　（別表）
EE71～CA		DISK命令ジャンプテーブル　（別表）
EECB～EF09	TRPTBL	ON割込みジャンプテーブル　（別表）
EF0A	RATEMP	ROM 5 をコールしたときのAcc, リターンしたときのAcc
EF0B, 0C	RMTEMP	ROM 5 をコールしたときのHL
EF0D	KANFLG	COPY 4 , 5 のフラグ （COPY 4 … 2 , COPY 5 … 3 ）
EF0E	S.INTM	ポートE 6 Hに出力したデータ （インタラプトマスク）
EF0F	FDIOFG	DMA FD使用中フラグ
EF10	DSKINT	FDイニシャライズ中フラグ
EF11	FD 5 PSN	DMA FDイニシャライズ中 0 になる
EF12, 13	SPMTRL	論理転送アドレス
EF14	DSKTMO	ミニフロッピー　TIME OUTまでの時間
EF15	FSTTNS	PC-8031- 2 W高速転送フラグ
EF16	CMDFL	DMA FD　転送ルーチン中はFF
EF17	PMTRC	DMA FD　R／Wスイッチ （WRITE＝ 2 , READ＝ 3 , INIT＝ 0 ）
EF18	PMTRD	DMA FD ドライブ番号, サーフェース番号 （bit 2 ：S, bit 1 , 0 ：D）
EF19	PMTRT	DMA FD　トラック番号
EF1A	PMTRS	DMA FD　セクタ番号
EF1B	PMTSC	DMA FD　セクタ数
EF1C, 1D	PMTRL	DMA FD　物理転送アドレス
EF1E～20	TMRL	DMA FD　タイマ
EF21	DMARW	DMACへのコマンド （READ＝40H, WRITE＝80H）
EF22	FCDRW	FDCへのコマンド （READ＝46H, WRITE＝45H）
EF23	RWERC	DMA FD　エラー番号 （READ＝87H, WRITE＝86H）
EF24		DMA FD　リトライ減算カウンタ
EF25		未使用
EF26	ST 0	ステータス 0 FDC Result phaseで読んだ値
EF27	ST 1	ステータス 1　　〃

アドレス	ラ ベ ル	機 能・用 途
EF28	ST2	ステータス2　FDC Result phaseで読んだ値
EF29	CCC	シリンダ番号　〃
EF2A	HH	ヘッド(サーフェス)番号　〃
EF2B	RR	セクタ番号　〃
EF2C	NN	セクタ内データ長　〃
EF2D~34	FDMFL	DMA FD ドライブ ステータス テーブル(1)
EF35~3C	FDFFL	DMA FD ドライブ ステータス テーブル(2)
EF3D	ERRFL	DMA FD エラー番号
EF3E	WAITF	インタラプト ウェイト フラグ
EF3F	TM	割込みレベル退避
EF40	TM1	FDC INT ステータス (ST0)
EF41~48	MRGP0	マージンデータテーブルポインタ
EF49	RUNBNF	BLOADのRフラグ
EF4A	NUMSC2	R／Wセクタ数
EF4B, 4C	FD0FL	(EF4B~EF5CはDMA FD用の定数) ドライブ ステータス テーブル アドレス
EF4D	MGPRT	マージンコントロール ポートアドレス
EF4E	IFPCK	インタフェイスボードチェックポートアドレス (bit0＝0…あり)
EF4F	FMOTOR	モーターコントロール ポートアドレス (PRE-COMPENSATIONのため)
EF50	MXTRK	片面あたりのトラック数
EF51	MDLTRK	まん中のトラック番号(このトラック以上ではPRE-COMPENSATIONを使う)
EF52	MXSCT	1トラックあたりのセクタ数
EF53	GAP3	GAP3の長さ(未使用)
EF54	FDCSB	FDC ステータスポートアドレス
EF55	FDCDB	FDC データポートアドレス
EF56	FDRDY	FD レディデータ (5インチ＝10H, 8インチ＝20H) F3HにOUTする
EF57	STPRT	Step rate time
EF58	HDLD	Head load time
EF59	DMAMD	DMAC モードデータ
EF5A	DMACH	DMAC チャネルアドレス
EF5B	DMAIO	DMAC ベースアドレス
EF5C	IFEN	インターフェイスイネーブルデータ F3HにOUTする。

アドレス	ラベル	機能・用途
EF5D	CURTYP	カレントドライブタイプ （0,1＝DMA，2＝SS，3＝DS）
EF5E		未使用
EF5F	DRVNUM	各ディスクタイプごとのドライブ番号
EF60	MAXFLP	DMA 8インチのドライブ数
EF61	MAXMIN	DMA 5インチのドライブ数
EF62	MAXINT	SS・DSのドライブ数
EF63	SGNBYT	SS・DSフラグ（SS＝0，DS＝80H）
EF64～6F	DRVTBL	ドライブタイプ対応表 （ドライブ番号→ドライブタイプ）
EF70		未使用
EF71	TRMXON	TERM　センドイネーブル
EF72	TRMHLF	TERM　HALF／FULL
EF73，74	TRMTXT	TERM　テキスト・ポインタ退避
EF75	LITFLG	TERM　リテラルフラグ（f・6）
EF76，77	FRETP	TERM　ポインタA
EF78，79	TPMEM	TERM　ポインタB
EF7A，7B	BTMEM	TERM　ポインタC
EF7C，7D	BOTPTR	TERM　ポインタD
EF7E	ROLFLG	TERM　ROLLバッファがあるかないかを示す。（1のときある）
EF7F	S.DIP	PORT 30H（IN）のCPL(ディップスイッチ1)
EF80		PORT 31H（IN）のCPL(ディップスイッチ2)
EF81	PENX	ライトペンX座標
EF82	PENY	ライトペンY座標
EF83，84	FSTPOS	1行入力最初のX，Y
EF85	LSTPOS	1行入力最後のX
EF86	CSRY	カーソル位置Y（1～）
EF87	CSRX	カーソル位置X（1～）
EF88	LINCNT	画面縦
EF89	LINWDT	画面横
EF8A，8B	ATRBUF	アトリビュートアドレス
EF8C	ATRNEW	セットするアトリビュートコード
EF8D	ATRCOL	アトリビュート桁

アドレス	ラベル	機能・用途
EF8E	ATRCNT	アトリビュート桁カウンタ
EF8F	LSTCHR	CRTに表示した文字
EF90	CNSDF2	ファンクションキー表示開始番号
EF91~98	GRPDOT	COPY　GVRAMコピー用バッファ
EF99		未使用
EF9A~B2	LINTAB	テキスト画面行継続コード （0：つながっている，0AH：^J）
EFB3		未使用
EFB4	F.EDIT	エディットモードフラグ
EFB5，B6	SCDTMP	行サーチテンポラリアドレス
EFB7，B8	SCDADR	行サーチ，サーチした行の1つ前の行のアドレス
EFB9	F.PINL	01＝エディット，FF＝1行入力 00＝ノーマル
EFBA，BB	HLPTXA	エラー位置を決めるため，A0Dからの読み込みルーチンでHLの位置をSAVEしておく。
EFBC，BD	HLPERA	エラー位置 （HELPでカーソルが移動するところ）
EFBE，BF	HLPERL	エラーのあったところの行番号
EFC0，C1	HLPBFA	中間コード→リスト形式にしたときのHLPERAに対応するエラー位置
EFC2，C3	HLPCSR	HLPBFAをプリントしたときのエラー位置のカーソル座標
EFC4，C5	WAITC0	(LINE) INPUT　WAIT文　待ち時間
EFC6~C8	WAITC1	(LINE) INPUT　WAIT用　DECカウンタ
EFC9~CC	TIMRC0	ON TIME$用　DECカウンタ
EFCD~D8	QUETAB	キューテーブル　（6バイト×2組）　EFCD~D2はキー入力バッファ用 EFCD　EFD3　…Put　オフセット CE　D4　…Get　オフセット CF　D5　…Back　Character D0　D6　…キューの長さ（2^n－1） D1,D2　D7,D8…キューアドレス
EFD9~F8	KIQADR	キー入力バッファ
EFF9	F.KSCN	キースキャンフラグ　（3＝押していない，FF＝押したまま，2＝新しくキーを押した）
EFFA	NKEYBT	新しく押されたキーのビットが1
EFFB，FC	NKEYAD	キーステータテーブル2 （E6DC~E6E7）用ポインタ

アドレス	ラ ベ ル	機 能・用 途
EFFD	KEYCOD	キーのコード (bit 2 ～ 0 ＝Data line No, bit 6 ～ 3 ＝ Port No.)
EFFE	ASCKEY	入力したキーのASCIIコード
EFFF	SHFBIT	キー入力シフトポートデータ
F000	OLDSHF	前回シフトポートデータ
F001	CAPLOK	CAPS LOCKフラグ
F002	SHFVAL	シフト番号 (0：ノーマル, 1＝SHIFT, 2＝CTRL, 3＝カナ, 4カナSHIFT, 5＝GRPH)
F003, 04	KEYSTR	ファンクションキーアドレス
F005	CASATR	カセット ファイル属性
F006	CASFL 2	カセット TPエラー disableフラグ (ヘッダサーチ中)
F007	CASFL 3	カセット STOPキーでNEWしないフラグ (ロード中は0)
F008	CASBAK	カセット Back character
F009	CASSPD	カセット ボーレート (FB＝1200ボー, FA＝600ボー)
F00A	F.CVFY	カセット ベリファイフラグ
F00B	COMXOF	TERM Xパラメータ有効フラグ&ステータス (^S, ^Q)
F00C	PFKYNM	ファンクションキー番号
F00D～12	TIMEB	時計用バッファ (BCD) 秒, 分, 時, 日, 月, 年, の順
F013	ONOFFF	PUT@文 条件または ONVAL, OFFVAL があるかどうかのフラグ
F014	ONVAL	PUT@文 フォアグラウンドカラーパレット番号
F015	OFFVAL	PUT@文 バックグラウンドカラーパレット番号
F016	ONMSK	PUT@文 Work
F017	OFFMSK	PUT@文 Work
F018	ONVADP	PUT@文 Work
F019	OFVADP	PUT@文 Work
F01A, 1B	GXPOS	スクリーン座標X
F01C, 1D	GYPOS	スクリーン座標Y
F01E	FORCLR	フォアグラウンド カラーパレット番号
F01F	BAKCLR	バックグラウンド カラーパレット番号
F020	BRDCLR	ボーダーカラー カラーコード
F021, 22	MAXDEL	GET@文 パターンの横のドット数, LINE文 横のドット数
F023, 24	MINDEL	GET@文 パターンの縦のドット数, LINE文 縦のドット数

アドレス	ラベル	機能・用途
F025, 26	LINSTL	LINE文　ラインスタイル
F027, 28	GRPACX	LP (Last Referenced Point) のスクリーン座標X
F029, 2A	GRPACY	LPのスクリーン座標Y
F02B, 2C	VXLEFT	ビューポート左上X
F02D, 2E	VXRGHT	ビューポート右下X
F02F, 30	VYLEFT	ビューポート左上Y
F031, 32	VYRGHT	ビューポート右下Y
F033, 34	CADDR	READ／WRITEするGVRAMのメモリアドレス
F035	CMASK	(CADDR)に書き込むデータパターン
F036	CURAM 1	GVRAM 1のマスクパターン
F037	CURAT 2	GVRAM 2のマスクパターン
F038	CURAT 3	GVRAM 3のマスクパターン
F039, 3A	MAXY	1つのGVRAMあたりの縦の座標の最大（＝199）
F03B, 3C	ARYADR	GET@, PUT@文　配列データポインタ
F03D	BYTCNT	GET@, PUT@文　Work
F03E	FINCNT	GET@, PUT@文　Work
F03F	INTCNT	GET@, PUT@文　Work
F040	INIMSK	GET@, PUT@文　Work
F041	FINMSK	GET@, PUT@文　Work
F042, 43	PUTACT	PUT@文サブルーチンアドレス (条件によって変わる)
F044	ATRBYT	PAINT文　領域色パレット番号, LINEの色パレット番号
F045	BRDATR	PAINT文　境界色パレット番号
F046	BRDAT 1	BRDATRのbit 0　ON／OFF
F047	BRDAT 2	BRDATRのbit 1　ON／OFF
F048	BRDAT 3	BRDATRのbit 2　ON／OFF
F049	PNTFLG	PAINT文　Work
F04A		未使用
F04B	BNKPRT	アクティブページ　(COPY文などのWork)
F04C, 4D	CASVEA	ペアレジスタ退避用
F04E	CSAVEM	レジスタ退避用
F04F, 50	TILBGN	PAINT文　タイルストリングの先頭のアドレス
F051	TILBFG	PAINT文　タイルペイントWork
F052, 53	TILEND	PAINT文　タイルストリングの終わりのアドレス＋1
F054, 55	SLCBGN	PAINT文　バックグラウンドストリングの先頭のアドレス
F056	TILFLG	PAINT文　タイルペイントを行うかどうかのフラグ

アドレス	ラ ベ ル	機 能・用 途
F057	TILLEN	(タイルストリングの長さ)￥M－1 (M：カラーモード3，B／Wモード1)
F058	TILNDX	PAINT文 タイルストリングカウンタ
F059～5B	TILBAK	バックグラウンドストリングデータ (8ドット分)
F05C～61		未使用
F062，63	PFRESZ	PAINT文で使えるフリーエリアの大きさ
F064，65	PSNLEN	PAINT文 サーチポイント･キューの長さ
F066，67	QUEINP	PAINT文 サーチポイント・キューの終わり
F068，69	QUEOUT	PAINT文 サーチポイント・キューの先頭
F06A	PDIREC	PAINT文 サーチ方向
F06B，6C	MOVCNT	PAINT文 Work
F06D，6E	SKPCNT	PAINT文 Work
F06F	LFPROG	PAINT文 Work
F070	RTPROG	PAINT文 Work
F071	PUTFLG	GET@，PUT@のフラグ (GET@…0，PUT@…80)
F072，73	ASPECT	CIRCLE文 (だ円率)＊256 (比率＝1…256，比率0.5or 2…128)
F074，75	CSTCNT	CIRCLE文 開始角度，終了角度のうち小さい方の値
F076，77	CENCNT	CIRCLE文 開始角度，終了角度のうち大きい方の値
F078，79	CRCSUM	CIRCLE文 Work
F07A	CPLOTF	開始角度，終了角度の大小を示すフラグ
F07B	CLINEF	円弧を書くかどうかのフラグ (X000000X)
F07C，7D	CNPNTS	CIRCLE文 半径＊sin(π/4) の値
F07E，7F	CPCNT8	CIRCLE文 Work
F080，81	CXOFF	CIRCLE文 Work
F082，83	CYOFF	CIRCLE文 Work
F084	CSCLXY	CIRCLE文 比率1以上かどうかのフラグ
F085，86	CPCNT	CIRCLE文 Work
F087	SCNMOD	スクリーンモード (0，1，2)
F088	SCNFLS	画面スイッチ (0，1，2，3)
F089	ACTPGE	READ／WRITE ページセレクトポート
F08A	PGECNT	ページ数(カラーモード3，B／Wモード1)
F08B	SCNPGE	アクティブ ページ セレクトポート (カラーモードでは5C)
F08C	DISPGE	ディスプレイページ
F08D，8E	VYMAX	縦のドット数 (640×400→400，640×200→200)

アドレス	ラベル	機能・用途
F08F	VIWDFG	ウィンドウフラグ WINDOW文が実行されると1になる。
F090, 91	VXDIFF	ビューポート　Sx_2-Sx_1
F092, 93	VYDIFF	ビューポート　Sy_2-Sy_1
F094~97	WXDIFF	ウィンドウ　Wx_2-Wx_1
F098~9B	WYDIFF	ウィンドウ　Wy_2-Wy_1
F09C~9F	FRX	VXDIFF／WXDIFF
F0A0~A3	FRY	VYDIFF／WYDIFF
F0A4~A7	GRPAXF	LPのワールド座標X
F0A8~AB	GRPAYF	LPのワールド座標Y
F0AC~AF	FTEMP	グラフィック処理用　Work
F0B0~B3	WXLEFT	ウィンドウ　左上　X
F0B4~B7	WXRGHT	ウィンドウ　右下　X
F0B8~BB	WYLEFT	ウィンドウ　左上　Y
F0BC~BF	WYRGHT	ウィンドウ　右下　Y
F0C0, C1	TPADR	ビューポート（Sx_1, Sy_1）のGVRAMメモリ・アドレス ＋80
F0C2	TOPPGE	ビューポート（Sx_1, Sy_1）のGVRAMセレクトポート
F0C3, C4	BOTADR	ビューポート（Sx_1, Sy_2）のGVRAMメモリアドレス
F0C5	BOTPGE	ビューポート（Sx_1, Sy_2）のGVRAMセレクトポート
F0C6, C7	LFMKAD	ビューポート（Sx_1, 0）のメモリアドレス
F0C8, C9	RHMKAD	ビューポート（Sx_2, 0）のメモリアドレス
F0CA	VIEWLM	ビューポート　左端の対応ビット
F0CB	VIEWRM	ビューポート　右端の対応ビット
F0CC, CD	LFTADR	PAINT文　サーチするドットラインの左端のメモリアドレス
F0CE, CF	RHTADR	PAINT文　サーチするドットラインの右端のメモリアドレス
F0D0~D2	SWTBAS	NEW ON 1でN-BASICへの切り換えに使う。
F0D3~F152	CASQUE	カセット入力バッファ
F153	INSISO	RS-232C　SI／SOフラグ（入力）
F154	OTSISO	RS-232C　SI／SOフラグ（出力）
F155~CD		RAM上に置かれているモニタサブルーチン
F1CE		モニタ　RADIX フラグ H or Q
F1CF		モニタ　ブレークポイントフラグ bit 0, bit 1
F1D0		モニタ　ブレークポイント1のデータ
F1D1, D2		モニタ　ブレークポイント1のアドレス
F1D3		モニタ　ブレークポイント2のデータ

アドレス	ラベル	機能・用途
F1D4, D5		モニタ　ブレークポイント2のアドレス
F1D6, D7		モニタ　Dコマンドのアドレス
F1D8, D9		モニタ　S, Eコマンドのアドレス
F1DA, DB		モニタ　Lコマンドのアドレス
F1DC, DD		モニタ　Aコマンドのアドレス
F1DE		モニタ　前に実行したコマンド
F1DF		モニタ　Eコマンドフラグ （プリンタへの出力をさせない）
F1E0		モニタ　1バイト数値入力フラグ
F1E1		モニタ　ROMセレクト
F1E2～E7		モニタ　ファイル名1
F1E8～ED		モニタ　ファイル名2
F1EE		未使用
F1EF		モニタ　READ／VERIFY　フラグ
F1F0		未使用
F1F1		モニタ　2バイト数値入力フラグ
F1F2, F3		モニタ　オフセットアドレス
F1F4		モニタ　プリンタスイッチ
F1F5, F6		モニタ　HL退避
F1F7, F8		モニタ　SP退避
F1F9		モニタ　テキスト ウィンドウ オフセットアドレスレジスタ退避
F1FA～FC		モニタ　HOOK（EDC9～）退避
F1FD～F216		モニタ　Xコマンド用レジスタ退避
F217～1D		モニタ　数値変換バッファ
F21E～FF		未使用
F300～1F		インタラプトベクトル
F320～C7		未使用
F3C8～FFF7		テキストVRAM
FFF8～FF		未使用

――N88-DISK-BASIC使用時のフックアドレステーブル――

アドレス	内		容
ECBB	JP	8A12	ディスク情報セット
ECBE	RET		
ECC1	JP	A736	何もしない
ECC4	JP	9BC5	ディスクSAVEルーチン
ECC7	RET		
ECCA	RET		
ECCD	JP	8E58	ドライブテーブルINIT,ID実行
ECD0	JP	9FFD	ディスクGET♯/PUT♯
ECD3	JP	91A5	EDIT行番号セットUP
ECD6	JP	91BE	EDIT行番号セットDOWN
ECD9	JP	9C51	ディスクLOADルーチン
ECDC	JP	A6E3	LINE INPUT♯ READ BACK
ECDF	RET		
ECE2	JP	9118	ROLL DOWNキー
ECE5	JP	90CC	ROLL UPキー
ECE8	JP	A08D	シーケンシャル出力
ECEB	RET		
ECEE	JP	A0D4	シーケンシャル入力
ECF1	RET		
ECF4	RET		
ECF7	RET		
ECFA	RET		
ECFD	RET		
ED00	RET		
ED03	RET		
ED06	JP	8E55	デバイス♯デフォルトセット
ED09	JP	8E9A	GETデバイス♯
ED0C	RET		
ED0F	JP	8F0A	シーケンシャル入力
ED12	RET		
ED15	RET		
ED18	RET		
ED1B	RET		
ED1E	RET		
ED21	RET		
ED24	JP	8DE5	1行入力後の処理
ED27	JP	8FA4	ディスク関数評価
ED2A	RET		

アドレス	内		容
ED2D	JP	8FBD	ファイル関数評価
ED30	RET		
ED33	RET		
ED36	RET		
ED39	RET		
ED3C	JP	8F06	LISTの先頭
ED3F	RET		
ED42	JP	91E3	1文字出力　カーソル補正
ED45	RET		
ED48	RET		
ED4B	RET		
ED4E	RET		
ED51	RET		
ED54	RET		
ED57	RET		
ED5A	JP	8F7C	倍精度関数評価
ED5D	RET		
ED60	RET		
ED63	RET		
ED66	JP	8FDD	エラーメッセージセット
ED69	RET		
ED6C	RET		
ED6F	JP	9B21	CLOSEディスクファイル
ED72	RET		
ED75	JP	8F41	式の評価
ED78	RET		
ED7B	RET		
ED7E	RET		
ED81	RET		
ED84	RET		
ED87	RET		
ED8A	RET		
ED8D	RET		
ED90	RET		
ED93	RET		
ED96	RET		
ED99	RET		
ED9C	RET		
ED9F	JP	AD48	プロテクトチェックI

アドレス	内		容
EDA2	JP	ACC1	SAVE暗号化
EDA5	JP	AD05	LOAD平文化
EDA8	JP	AD51	プロテクトチェックII
EDAB	JP	9A01	OPENディスクファイル
EDAE	RET		
EDB1	JP	994B	ドライブポインタセット
EDB4	RET		
EDB7	RET		
EDBA	RET		
EDBD	RET		
EDC0	RET		
EDC3	RET		
EDC6	RET		
EDC9	JP	8E4B	CHAINキャンセル
EDCC	RET		
EDCF	RET		
EDD2	RET		
EDD5	RET		
EDD8	RET		
EDDB	RET		
EDDE	RET		
EDE1	RET		
EDE4	RET		
EDE7	RET		
EDEA	RET		
EDED	RET		
EDF0	RET		
EDF3	RET		
EDF6	JP	908B	ROM Ver1,0のためのパッチ
EDF9	RET		
EDFC	RET		
EDFF	JP	9977	マウント確認
EE02	RET		
EE05	RET		
EE08	RET		
EE0B	RET		

——N88-DISK-BASIC使用時のI/O処理ルーチン　ジャンプテーブル——

アドレス	内		容	
EE0E	JP	4DC1	OPEN	
EE11	JP	4DC1	CLOSE	
EE14	JP	4DC1	PUT/GET	
EE17	JP	4DC1	OUTPUT	
EE1A	JP	4DC1	INPUT	
EE1D	JP	4DC1	LOC	COM2,3
EE20	JP	4DC1	LOF	
EE23	JP	4DC1	EOF	
EE26	JP	4DC1	FPOS	
EE29	JP	4DC1	READ BACK	
EE2C	JP	4DC1	WIDTH#	
EE2F	JP	9043	OPEN	
EE32	JP	483D	CLOSE	
EE35	JP	905F	PUT/GET	
EE38	JP	9051	OUTPUT	
EE3B	JP	0B06	INPUT	
EE3E	JP	0B06	LOC	SCRN
EE41	JP	0B06	LOF	
EE44	JP	0B06	EOF	
EE47	JP	0B06	FPOS	
EE4A	JP	0B06	READ BACK	
EE4D	JP	0B06	WITH#	
EE50	JP	9001	OPEN	
EE53	JP	483D	CLOSE	
EE56	JP	9018	PUT/GET	
EE59	JP	0B06	OUTPUT	
EE5C	JP	900C	INPUT	
EE5F	JP	903C	LOC	KYBD
EE62	JP	0B06	LOF	
EE65	JP	0B06	EOF	
EE63	JP	0B06	FPOS	
EE6B	JP	9032	READ BACK	
EE6E	JP	0B06	WIDTH#	

——N88-DISK-BASIC使用時のDISK命令ジャンプテーブル——

アドレス	内		容
EE71	JP	A72B	Disk I/O error
EE74	JP	A72E	Disk offline
EE77	JP	8EC3	DSKF
EE7A	JP	A98C	CHAIN
EE7D	JP	88E9	SEARCH
EE80	JP	A875	WHILE
EE83	JP	A89F	WEND
EE86	JP	AC75	WRITE
EE89	JP	A916	CALL
EE8C	JP	9DAD	SET
EE8F	JP	99B6	NAME
EE92	JP	9BAD	KILL
EE95	JP	A185	DSKI$
EE98	JP	A1C4	DSKO$
EE9B	JP	9F2F	FILES
EE9E	JP	9F2A	LFILES
EEA1	JP	4DC1	WBYTE
EEA4	JP	4DC1	RBYTE
EEA7	JP	4DC1	POLL
EEAA	JP	4DC1	ISET
EEAD	JP	4DC1	IRESET
EEB0	JP	4DC1	STATUS
EEB3	JP	4DC1	STATUS
EEB6	JP	4DC1	CMD
EEB9	JP	4DC1	IEEE
EEBC	JP	8803	ROLL
EEBF	JP	A7A8	BLOAD
EEC2	JP	A73D	BSAVE
EEC5	JP	AC72	COMMON
EEC8	JP	89B3	ATN

——N88-DISK-BASIC使用時のON割込みアドレステーブル——

アドレス	内	容
EECB	ON STOP	3バイトで1組 1バイト目 bit 7 6 5 4 3 2 1 0 bit 0: ON bit 1: STOP bit 2: リクエストあり 2，3バイト目 インタラプトルーチンのアドレス
EECE	ON COM1	
EED1	ON COM2	
EED4	ON COM3	
EED7	ON PEN	
EEDA	ON TIME$	
EEDD	ON HELP	
EEE0	ON KEY(1)	
EEE3	ON KEY(2)	
EEE6	ON KEY(3)	
EEE9	ON KEY(4)	
EEEC	ON KEY(5)	
EEEF	ON KEY(6)	
EEF2	ON KEY(7)	
EEF5	ON KEY(8)	
EEF8	ON KEY(9)	
EEFB	ON KEY(10)	
EEFE		
EF01		
EF04		
EF07		

付録6　I/Oポート一覧表

ポートアドレス		内　　　　容
00H ～ 0BH	0 ～ 11	キーボード　(IN)

ポートアドレス		内　　容
10H	16	プリンタ・データ出力ポート（OUT） プリンタ用バスにデータを出力します。
10H	16	カレンダークロック（μPD1990）用出力ポート（OUT） μPD1990にコマンドやデータを出力します。
20H	32	USART（μPD8251）データポート（IN/OUT）
21H	33	USART（μPD8251）コントロールポート（IN/OUT） モードインストラクションの定義（OUT）

	7	6	5	4	3	2	1	0
OUT	PDB7	PDB6	PDB5	PDB4	PDB3	PDB2	PDB1	PDB0

	7	6	5	4	3	2	1	0
OUT					CDO	C2	C1	C0

bit	内　　容	
0 1 2	C0 C1 C2	μPD1990へのコマンド出力
3	CDO	μPD1990へのデータ出力

非同期モード

	7	6	5	4	3	2	1	0	
MSB	S_2	S_1	EP	PEN	L_2	L_1	B_2	B_1	LSB

ボーレート

0	1	0	1
0	0	1	1
同期モード		X16	X64

キャラクタ長

0	1	0	1
0	0	1	1
5ビット	6ビット	7ビット	8ビット

パリティイネーブル
1：イネーブル
0：ディスエイブル

パリティ指定
1：偶　数
0：奇　数

ストップビット数

0	1	0	1
0	0	1	1
無　効	1ビット	$1\frac{1}{2}$ビット	2ビット

ポートアドレス
内　容
同期モード
SCS ESC EP PEN L2 L1 0 0
キャラクタ長
0 1 0 1
0 0 1 1
5ビット 6ビット 7ビット 8ビット
パリティイネーブル
1：イネーブル
0：ディスエイブル
パリティ指定
1：偶　数
0：奇　数
外部同期検出
1：SYNDET＝入力
0：SYNDET＝出力
単一キャラクタ同期
1：単一SYNCキャラクタ
0：ダブルSYNCキャラクタ
コマンドインストラクションの定義（OUT）
EH IR RTS ER SBRK RxE DTR TxEN
送信イネーブル
1：イネーブル
0：ディスエイブル
データ・ターミナルレディ
1：Date Terminal ReadyをONにする。
0：Date Terminal ReadyをOFFにする。
受信イネーブル
1：イネーブル
0：ディスエイブル
センドブレイク・キャラクタ
1：ブレイクキャラクタの送信
0：通常動作
エラーリセット
1：エラーフラグ(PE, OE, FE)をリセット
0：NO OPERATION
センド要求
1：Request to Send をONにする。
0：Request to Send をOFFにする。
内部リセット
1：8251をモード・インストラクションフォーマットへもどす。
0：NO OPERATION
HUNT
1：SYNCキャラクタ検出を始める。
0：NO OPERATION

ポートアドレス
内容
ステータスの読み出し（IN）
DSR
SYN DET
FE
OE
PE
TxE
Rx RDY
Tx RDY
送信レディ
1：レディ
0：ビジー
受信レディ
1：レディ
0：ビジー
送信バッファエンプティ
1：エンプティ
0：フル
パリティエラー
1：パリティエラー発生
0：エラーなし
オーバーランエラー
1：オーバーランエラー発生
0：エラーなし
フレミングエラー
1：フレミングエラー発生
0：エラーなし
SYNCキャラクタ検出
1：SYNCキャラクタ検出
0：検出なし
Data Set Ready
1：Data Set Ready ON
0：Data Set Ready OFF
（Data Set Ready端子の状態をモニタできます。）

ポートアドレス		内　　　容
30H	48	システムコントロールポート(1)

OUT

7	6	5	4	3	2	1	0
／	／	BS2	BS1	$\overline{\text{MTON}}$	CDS	$\overline{\text{COLOR}}$	$\overline{40}$

bit		内　容
0	$\overline{40}$	CRTディスプレイFORMATコントロール 0：40文字モード 1：80文字モード
1	$\overline{\text{COLOR}}$	CRTディスプレイモードコントロール 0：カラーモード 1：モノクロモード
2	CDS	CMTキャリアコントロール 0：スペース 1：マーク
3	MTON	CMTのモータコントロール 0：OFF 1：ON
4 5	BS2 BS1	USARTのチャンネルコントロール BS2　BS1 0　0　：CMT600bps 0　1　：CMT1200bps 1　0　：RS-232C（非同期） 1　1　：RS-232C（同期）

ポートアドレス			内　　　容
			ディップ・スイッチ，汎用入力ポート(1)

	7	6	5	4	3	2	1	0
IN	UIP1	UIP0	SW1-6	SW1-5	SW1-4	SW1-3	SW1-2	SW1-1

bit	内		容
0	SW1-1	0：	N-BASIC
		1：	N_{88}-BASIC
1	SW1-2	0：	ターミナルモード
		1：	BASIC
2	SW1-3	0：	80文字/行
		1：	40文字/行
3	SW1-4	0：	25行/画面
		1：	20行/画面
4	SW1-5	0：	Sパラメータ有効
		1：	Sパラメータ無効
5	SW1-6	0：	DELコードを処理する。
		1：	DELコードを無視する。
6	UIP0	汎用入力ポート	
7	UIP1		

ポートアドレス		内　　容
31H	49	システムコントロールポート(2)

	7	6	5	4	3	2	1	0
OUT			25LINE	HCOLOR	GRPH	RMODE	MMODE	200LINE

bit		内　容
0	200LINE	ハイスピードCRTモードにおけるグラフィックモードのコントロール 0：640×400×1 1：640×200×3
1	MMODE	RAMモードのコントロール 0：ROM,RAMモード 1：64KRAMモード
2	RMODE	ROMモードのコントロール 0：N_{88}-BASICモード（ROM3,4） 1：N-BASICモード（ROM1,2）
3	GRPH	グラフィックディスプレイモード 0：グラフィック画面を表示しない。 1：　　　〃　　　表示する。
4	HCOLOR	カラーグラフィックディスプレイモード 0：モノクロモード 1：カラーモード
5	25LINE	ハイスピードCRTモードにおけるLINE/FRAMEコントロール 0：20LINEモード 1：25LINEモード

ポートアドレス		内　　容

ディップ・スイッチ，入力ポート(2)

	7	6	5	4	3	2	1	0
IN	UIP3	UIP2	SW2-6	SW2-5	SW2-4	SW2-3	SW2-2	SW2-1

bit	内	容	
0	SW2-1	0：	パリティ有り
		1：	パリティ無し
1	SW2-2	0：	偶数(EVEN)パリティ
		1：	奇数(ODD)パリティ
2	SW2-3	0：	8ビット
		1：	7ビット
3	SW2-4	0：	ストップ・ビット＝2
		1：	ストップ・ビット＝1
4	SW2-5	0：	Xパラメータ有効
		1：	Xパラメータ無効
5	SW2-6	0：	半二重
		1：	全二重
6	UIP2	汎用入力ポート	
7	UIP3		

ポートアドレス		内　　容
40H	64	ストローブポート

	7	6	5	4	3	2	1	0
OUT		UOP0	BEEP	FLASH	$\overline{\text{CLDS}}$	CCK	CSTB	$\overline{\text{PSTB}}$

bit		内　　容
0	$\overline{\text{PSTB}}$	プリンタへのストローブ信号 0：アクティブ 1：インアクティブ
1	CSTB	カレンダクロックへのストローブ信号 0：インアクティブ 1：アクティブ
2	CCK	カレンダクロックへのシフトクロック 0：ノーマル 1：クロックON
3	$\overline{\text{CLDS}}$	CRTコントロール回路への同期パルス 0：インアクティブ 1：アクティブ
4	FLASH	フラッシングモードのコントロール 0：ノーマルモード 1：フラッシングモード
5	BEEP	ブザーのコントロール 0：BEEP OFF 1：BEEP ON
6	UOP0	汎用出力ポート

ポートアドレス		内　　容

IN	7	6	5	4	3	2	1	0
	／	／	VRTC	CDI	$\overline{\text{EXTON}}$	DCD	$\overline{\text{SHG}}$	BUSY

bit		内　　容
0	BUSY	プリンタからのBUSY信号 0：READY 1：BUSY
1	$\overline{\text{SHG}}$	ハイレゾリューショングラフィックモード信号 0：ハイレゾモード 1：ノーマルモード
2	DCD	SIOのデータキャリアディテクト信号 0：キャリア入力なし 1：キャリア入力あり
3	$\overline{\text{EXTON}}$	ミニディスクユニット接続信号 0：接続されている 1：接続されていない
4	CDI	カレンダクロックからのデータ入力
5	VRTC	CRTCからの垂直帰線信号 0：表示，水平帰線サイクル 1：垂直帰線サイクル

ポートアドレス		内　　　容
50H	80	CRTC（μPD3301）パラメータ　（OUT）
51H	81	CRTC（μPD3301）コマンド　（OUT）
52H	82	ボーダーカラー，バックグラウンドカラー制御（OUT）

	7	6	5	4	3	2	1	0
OUT	／	BGG	BGR	BGB	／	RG	RR	RB

bit	内	容
0	RB	ボーダーカラー　BLUE
1	RR	〃　RED
2	RG	〃　GREEN
4	BGB	バックグラウンドカラー　BLUE
5	BGR	〃　RED
6	BGG	〃　GREEN

ポートアドレス		内　　　容
53H	83	画面の重ね合わせの制御（OUT）

	7	6	5	4	3	2	1	0
OUT	／	／	／	／	G2DS	G1DS	G0DS	TXTDS

bit	内	容
0	TXTDS	TEXT画面の表示 0：表示する 1：表示しない
1	G0DS	G-VRAM0の表示 0：表示する 1：表示しない
2	G1DS	G-VRAM1の表示 0：表示する 1：表示しない
3	G2DS	G-VRAM2の表示 0：表示する 1：表示しない

ポートアドレス		内　　　容
54H	84	カラーパレット 0 の制御（OUT）
55H	85	カラーパレット 1 の制御（OUT）
56H	86	カラーパレット 2 の制御（OUT）

ポートアドレス		内　　容
57H	87	カラーパレット 3 の制御（OUT）
58H	88	カラーパレット 4 の制御（OUT）
59H	89	カラーパレット 5 の制御（OUT）
5AH	90	カラーパレット 6 の制御（OUT）
5BH	91	カラーパレット 7 の制御（OUT）

	7	6	5	4	3	2	1	0
OUT						PG	PR	PB

bit		内　　容
0	PB	カラーパレット　BLUE
1	PR	カラーパレット　RED
2	PG	カラーパレット　GREEN

ポートアドレス		内　　容
5CH		G VRAMステータス (IN)

	7	6	5	4	3	2	1	0
IN						G2	G1	G0

bit		内　　容
0	G0	G-VRAM 0 ステータス
1	G1	G-VRAM 1 ステータス
2	G2	G-VRAM 2 ステータス

ポートアドレス		内　　容
5CH	92	G-VRAM0　選択　（OUT）
5DH	93	G-VRAM1　選択　（OUT）
5EH	94	G-VRAM2　選択　（OUT）
5FH	95	メインRAM　選択　（OUT）

ポートアドレス		内　容
60H ～ 68H	96 ～ 104	DMAC(μPD8257)コントロールポート 60H～67H：OUT 68H：IN/OUT

PD8257レジスタ選択表

ポート		レジスタ	バイト	アドレス入力 A_3	A_2	A_1	A_0	F/L	双方向データバス（※） D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
60H	96	CH-0 DMAアドレス	下位	0	0	0	0	0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
			上位	0	0	0	0	1	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8
61H	97	CH-0 ターミナルカウント	下位	0	0	0	1	0	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0
			上位	0	0	0	1	1	R_d	W_r	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8
62H	98	CH-1 DMAアドレス	下位	0	0	1	0	0	チャネル0に同じ							
			上位	0	0	1	0	1								
63H	99	CH-1 ターミナルカウント	下位	0	0	1	1	0								
			上位	0	0	1	1	1								
64H	100	CH-2 DMAアドレス	下位	0	1	0	0	0								
			上位	0	1	0	0	1								
65H	101	CH-2 ターミナルカウント	下位	0	1	0	1	0								
			上位	0	1	0	1	1								
66H	102	CH-3 DMAアドレス	下位	0	1	1	0	0								
			上位	0	1	1	0	1								
67H	103	CH-3 ターミナルカウント	下位	0	1	1	1	0								
			上位	0	1	1	1	1								
68H	104	モードセット(プログラム時)		1	0	0	0	0	AL	TCS	EW	RP	EN3	EN2	EN1	EN0
		ステータス(読み出し時)	——	1	0	0	0	0	0	0	0	UP	TC3	TC2	TC1	TC0

CH·0　　5インチDMAタイプディスクユニット
CH·1　　8インチディスクユニット
CH·2　　CRTC

※A_0～A_{15}：DMA開始アドレス　　C_0～C_{13}：TCの値（N−1）
R_dとW_r：ベリファイ（００），ライト（０１），リード（１０）
AL：オートロード　　TCS：TCストップ
EW：拡張ライト　　RP：回転優先
EN３～０：チャネルイネーブル　　UP：UPDATEフラグ
TC３～０：TCステータスビット

ポートアドレス		内　容
70H	112	TEXT WINDOWアドレスのオフセットアドレスレジスタ（IN/OUT）

	7	6	5	4	3	2	1	0
OUT IN	AP15	AP14	AP13	AP12	AP11	AP10	AP9	AP8

ポートアドレス		内容
71H	113	4 th ROMコントロール 4 th ROMを制御します。

	7	6	5	4	3	2	1	0
OUT IN	$\overline{\text{4th}}$ ROM8	$\overline{\text{4th}}$ ROM7	$\overline{\text{4th}}$ ROM6	$\overline{\text{4th}}$ ROM5	$\overline{\text{4th}}$ ROM4	$\overline{\text{4th}}$ ROM3	$\overline{\text{4th}}$ ROM2	$\overline{\text{4th}}$ ROM1

ポートアドレス		内容
78H	120	TEXT WINDOWアドレスのオフセットアドレスレジスタを1増す。
E4H	228	割込みコントローラ（μPD8214）カレントステータス出力ポート（OUT）

	7	6	5	4	3	2	1	0
OUT	/	/	/	/	$\overline{\text{SGS}}$	$\overline{B_2}$	$\overline{B_1}$	$\overline{B_0}$

bit		内容
0	$\overline{B_0}$	カレントインタラプトレベル
1	$\overline{B_1}$	
2	$\overline{B_2}$	
3	$\overline{\text{SGS}}$	カレントステータスレジスタのコントロール

ポートアドレス		内容
E6H	230	割込みマスクフラグ（OUT）

	7	6	5	4	3	2	1	0
OUT	/	/	/	/	/	$\overline{\text{RTMF}}$	$\overline{\text{VRMF}}$	$\overline{\text{RXMF}}$

bit		内容
0	$\overline{\text{RTMF}}$	リアルタイム割込みのマスクフラグ
1	$\overline{\text{VRMF}}$	VRTC割込みのマスクフラグ
2	$\overline{\text{RXMF}}$	RxRDY割込みのマスクフラグ

ポートアドレス		内容
E8H	232	（OUT）漢字ROMアドレスの下位8 bit指定 （IN）　漢字フォントの読み出し（下位8 bit）
E9H	233	（OUT）漢字ROMアドレスの上位8 bit指定 （IN）　漢字フォントの読み出し（上位8 bit）
EAH	234	（OUT）漢字ROMの読み出し開始
EBH	235	（OUT）漢字ROMの読み出し終了
F3H	243	DMAタイプ ディスクユニット インターフェイスセレクトポート(OUT)
F4H ～ F7H	244 247	DMAタイプ 8インチディスクユニット制御ポート

<table>
<tr><th colspan="2">ポートアドレス</th><th>内　　容</th></tr>
<tr><td>F8H
～
FBH</td><td>248

251</td><td>DMAタイプ 5インチディスクユニット制御ポート

F4H	F8H	インターフェースボードチェック（IN）bit 0 モーターコントロール（OUT）PRE-COMPENSATIONのため
F5H	F9H	マージンコントロール（OUT）
F6H	FAH	FDCステータス（IN）
F7H	FBH	FDCデータ・レジスタ（IN/OUT）

</td></tr>
<tr><td>FCH
～
FFH</td><td>252

255</td><td>ミニディスクユニット制御ポート（μPD8255）
FCH：PORT A（IN）入力データ
FDH：PORT B（OUT）出力データ
FEH：PORT C（IN/OUT）ハンドシェイク
FFH：コントロールポート（OUT）</td></tr>
</table>

付録7　コマンド、ステートメント、関数処理アドレス一覧表

[[STATEMENT]]	MAIN	ROM4	DISK
AUTO ················	0DD5	----	----
BEEP ················	3EB4	----	----
BLOAD ···············	EEBF	----	A7A8
BSAVE ···············	EEC2	----	A73D
CALL ················	EE89	----	A916
CHAIN ···············	EE7A	----	A98C
CIRCLE ··············	6ECE	79D6	----
CLEAR ···············	522E	----	----
CLOSE ···············	4B04	----	----
CLS ·················	71B5	----	----
CMD ·················	EEB6	----	4DC1
COLOR ···············	6EC6	6878	----
COLOR@ ··············	----	6927	----
COLOR[1] ············	----	6882	----
COLOR[2] ············	----	68EC	----
COM ON/OFF/STOP ····	7D71	----	----
COMMON ··············	EEC5	----	AC72
CONSOLE ·············	7071	----	----
CONT ················	5140	----	----
COPY ················	6EA6	7053	----
DATA ················	0C77	----	----
DATE$ ···············	721C	----	----
DEF ·················	15D7	----	----
DEF FN ············	15DB	----	----
DEF USR ···········	15CC	----	----
DEFDBL ··············	0ACD	----	----
DEFINT ··············	0AC7	----	----
DEFSNG ··············	0ACA	----	----
DEFSTR ··············	0AC4	----	----
DELETE ··············	1B40	----	----
DIM ·················	5AC5	----	----
DSKO$ ···············	EE98	----	A1C4
EDIT ················	657B	----	----
ELSE ················	0C79	----	----
END ·················	50E5	----	----
ERASE ···············	519C	----	----
ERROR ···············	0DCA	----	----
FIELD ···············	4A5C	----	----
FILES ···············	EE9B	----	9F2F
FOR ·················	08BF	----	----
GET ·················	7198	----	----
GET# ··············	4B42	----	----
GET@ ··············	71A2	7C33	----
GOSUB ···············	0BBF	----	----
GOTO ················	0BF9	----	----
HELP ON/OFF/STOP ···	72AB	----	----
IF ··················	0E05	----	----
INPUT ···············	102D	----	----
INPUT ·············	1034	----	----
INPUT# ············	1020	----	----
INPUT WAIT ·······	1034	----	----

[[STATEMENT]]	MAIN	ROM4	DISK
IRESET ··············	EEAD	----	4DC1
ISET ················	EEAA	----	4DC1
KEY ·················	6EFA	742E	----
KEY ···············	----	7443	----
KEY LIST ··········	----	73DF	----
KEY ON/OFF/STOP ··	----	7437	----
KILL ················	EE92	----	9BAD
LET ·················	0C9C	----	----
LFILES ··············	EE9E	----	9F2A
LINE ················	0FAA	----	----
LINE ··············	6EAE	7E8B	----
LINE INPUT ········	0FB9	----	----
LINE INPUT# ·······	4CC1	----	----
LINE INPUT WAIT ··	0FB9	----	----
LIST ················	18D9	----	----
LLIST ···············	18D4	----	----
LOAD ················	4854	----	----
LOCATE ··············	714E	----	----
LPRINT ··············	0E4C	----	----
LSET ················	49AB	----	----
MERGE ···············	4855	----	----
MID$ ················	585A	----	----
MON ·················	E826	----	----
MOTOR ···············	7F30	----	----
NAME ················	EE8F	----	99B6
NEW ·················	77DD	----	----
NEW ···············	4F00	----	----
NEW ON ············	77E0	----	----
NEXT ················	52BD	----	----
ON ··················	0D01	----	----
ON ERROR GOTO ····	0D05	----	----
ON GOSUB/GOTO ····	0D73	----	----
ON XX GOSUB ······	0D34	----	----
OPEN ················	4798	----	----
OPTION BASE ·········	1C89	----	----
OUT ·················	17FA	----	----
PAINT ···············	6EDA	7674	----
PEN ON/OFF/STOP ····	72C8	----	----
POINT ···············	6EB2	6E25	----
POKE ················	1B84	----	----
POLL ················	EEA7	----	4DC1
PRESET ··············	6E96	7DFB	----
PRINT ···············	0E54	----	----
PSET ················	6E9A	7E00	----
PUT ·················	71A6	----	----
PUT# ··············	4B41	----	----
PUT@ ··············	71B0	7C33	----
RANDOMIZE ···········	1CD1	----	----
RBYTE ···············	EEA4	----	4DC1
READ ················	10F9	----	----
REM ·················	0C79	----	----
RENUM ···············	6F0E	75DD	----
RESTORE ·············	50A5	----	----
RESUME ··············	0D8D	----	----
RETURN ··············	0C41	----	----
ROLL ················	6ECA	EEBC	8803
RSET ················	49AA	----	----
RUN ·················	0B7C	----	----

SAVE	48A3	----	----
SCREEN	6EDE	698F	----
SET	EE8C	----	9DAD
STATUS	EEB0	----	4DC1
STOP	50CA	----	----
STOP	50CD	----	----
STOP ON/OFF/STOP	72B0	----	----
SWAP	515E	----	----
TERM	7367	----	----
TIME$	7279	----	----
TIME$ ON/OFF/STOP	7279	----	----
TIME$=	728F	----	----
TROFF	5159	----	----
TRON	5158	----	----
VIEW	6EBA	6AC6	----
WAIT	1800	----	----
WBYTE	EEA1	----	4DC1
WEND	EE83	----	A89F
WHILE	EE80	----	A875
WIDTH	181A	----	----
WIDTH	1875	----	----
WIDTH#	1847	----	----
WIDTH LPRINT	1866	----	----
WIDTH[DEV]	1828	----	----
WINDOW	6ED6	6C55	----
WRITE	EE86	----	AC75

[[FUNCTION]]	MAIN	ROM4	DISK
ABS	20A0	----	----
ASC	5704	----	----
ATN	EEC8	----	89B3
ATTR$	----	----	9E1D
CDBL	223E	----	----
CHR$	5714	----	----
CINT	21A0	----	----
COS	2F8B	----	----
CSNG	2214	----	----
CSRLIN	3F31	----	----
CVD	4AC0	----	----
CVI	4ABA	----	----
CVS	4ABD	----	----
DATE$	6F02	----	----
DSKF	EE77	----	8EC3
DSKI$	----	----	A185
EOF	4C51	----	----
ERL	13A2	----	----
ERR	1394	----	----
EXP	2E6E	----	----
FIX	2286	----	----
FN	1600	----	----
FPOS	4C62	----	----
FRE	58E4	----	----
HEX$	54C6	----	----
IEEE	EEB9	----	4DC1
INKEY$	5AA3	----	----
INP	17E5	----	----
INPUT$	4BAC	----	A104
INSTR	57D7	----	----
INT	2295	----	----

LEFT$	575A	----	----
LEN	56F8	----	----
LOC	4C2F	----	----
LOF	4C40	----	----
LOG	1F10	----	----
LPOS	1581	----	----
MAP	6E9E	6D29	----
MID$	5793	----	----
MKD$	4AA7	----	----
MKI$	4AA1	----	----
MKS$	4AA4	----	----
NOT	1512	----	----
OCT$	54C1	----	----
PEEK	1B7A	----	----
PEN	6EF2	72ED	----
POINT	6EC2	6DC0	----
POS	1586	----	----
RIGHT$	578A	----	----
RND	2F1A	----	----
SEARCH	EE7D	----	88E9
SGN	20B3	----	----
SIN	2F91	----	----
SPACE$	5741	----	----
SQR	2E05	----	----
STATUS	EEB3	----	4DC1
STR$	54CB	----	----
STRING$	5722	----	----
TAN	302C	----	----
TIME$	6EFE	752A	----
USR	158F	----	----
VAL	57B4	----	----
VARPTR	13B0	----	----
VIEW	6EE2	6CA8	----
WINDOW	6ED2	6CD5	----

付録8　コントロールコード一覧表

16進	10進	対応するキー	N−BASIC	N_{88}−BASIC
01	1	CTRL−A		ヘルプキーと同じ
02	2	CTRL−B	1つ前のワードへ戻る	(N−BASICと同じ)
03	3	CTRL−C	実行の中断（STOPの時）	実行の中断
04	4	CTRL−D		カーソル位置から1ワードを削除
05	5	CTRL−E	カーソル位置から後を消す	(N−BASICと同じ)
06	6	CTRL−F		1つ先のワードへ進む
07	7	CTRL−G	スピーカを鳴らす	(N−BASICと同じ)
08	8	CTRL−H	カーソル位置の左側の文字を削除する	(N−BASICと同じ)
09	9	CTRL−I	水平タブ（8文字毎）	(N−BASICと同じ)
0A	10	CTRL−J	行を2つに分ける	ラインフィード、インサートモードで2行に分割
0B	11	CTRL−K	ホームポジション	(N−BASICと同じ)
0C	12	CTRL−L	テキスト画面クリア	(N−BASICと同じ)
0D	13	CTRL−M	キャリッジリターン	(N−BASICと同じ)
0E	14	CTRL−N	1つ先のワードへ進む	
0F	15	CTRL−O	ESCの後に押すことによりN_{88}−BASICと同じ働きを行なう	画面の表示を無効にする
12	18	CTRL−R	カーソル位置から右側を1文字分右へずらす。	インサートモードにする。
13	19	CTRL−S		実行を一時定止する
15	21	CTRL−U		1行キャンセル
18	24	CTRL−X		カーソルを行の最後に移す
1B	27	ESC	実行を一時停止する	
1C	28	→	カーソルを右へ移動	(N−BASICと同じ)
1D	29	←	〃　左　〃	(N−BASICと同じ)
1E	30	↑	〃　上　〃	(N−BASICと同じ)
1F	31	↓	〃　下　〃	(N−BASICと同じ)

付録9　エラーメッセージ一覧表

DECI	HEX	-ERROR MESSAGE-
1	01	NEXT without FOR
2	02	Syntax error
3	03	RETURN without GOSUB
4	04	Out of DATA
5	05	Illegal function call
6	06	Overflow
7	07	Out of memory
8	08	Undefined line number
9	09	Subscript out of range
10	0A	Duplicate Definition
11	0B	Division by zero
12	0C	Illegal direct
13	0D	Type mismatch
14	0E	Out of string space
15	0F	String too long
16	10	String formula too complex
17	11	Can´t continue
18	12	Undefined user function
19	13	No RESUME
20	14	RESUME without error
21	15	Unprintable error
22	16	Missing operand
23	17	Line buffer overflow
24	18	?
25	19	?
26	1A	FOR Without NEXT
27	1B	Tape read ERROR
28	1C	?
29	1D	WHILE without WEND
30	1E	WEND without WHILE
31	1F	duplicate label
32	20	undefined label
33	21	Feature not available
50	32	FIELD overflow
51	33	Internal error
52	34	Bad file number
53	35	File not found
54	36	File already open
55	37	Input past end
56	38	Bad file name
57	39	Direct statement in file
58	3A	Sequential after PUT
59	3B	Sequential I/O only
60	3C	File not OPEN
61	3D	File write protected
62	3E	Disk offline
63	3F	Disk not mounted
64	40	Disk I/O error
65	41	File already exists
66	42	?
67	43	Disk already mounted
68	44	Disk full
69	45	Bad allocation table
70	46	Bad drive number

```
71   47  Bad track/sector
72   48  Deleted record
73   49  Rename across disks
```

N_{88}-ROM-BASIC，N_{88}-DISK-BASICエラーメッセージ対応表

```
AO  54  File already open
BN  52  Bad file number
BO  23  Line buffer overflow
BS   9  Subscript out of range
CF  60  File not OPEN
CN  17  Can't continue
DD  10  Duplicate Definition
DS  57  Direct statement in file
DU  31  duplicate label
EF  55  Input past end
FC   5  Illegal function call
FF  53  File not found
FN  26  FOR Without NEXT
FO  50  FIELD overflow
ID  12  Illegal direct
IE  51  Internal error
LS  15  String too long
MO  22  Missing operand
NA  33  Feature not available
NF   1  NEXT without FOR
NM  56  Bad file name
NR  19  No RESUME
OD   4  Out of DATA
OM   7  Out of memory
OS  14  Out of string space
OV   6  Overflow
RG   3  RETURN without GOSUB
RW  20  RESUME without error
SN   2  Syntax error
SP  58  Sequential after PUT
SQ  59  Sequential I/O only
ST  16  String formula too complex
TM  13  Type mismatch
TP  27  Tape read ERROR
UE  21  Unprintable error
UF  18  Undefined user function
UL   8  Undefined line number
UN  32  undefined label
WE  30  WEND without WHILE
WH  29  WHILE without WEND
/0  11  Division by zero
```

付録10　プリンタ機能一覧表(PC8821/22, PC8023)

分　　類	ニーモニック	HEXコード	機　　能	PC-8821/8822	PC-8023(C)
印字指令	CR	0D	バッファのデータを印字	○	○
改行	LF	0A	1行送り	○	○
垂直タブ	VT	0B	多行送り	○	○
フォームフィード	FF	0C	改ページ	○	○
拡　　大	SO	0E	拡大指令（8 bit）	○	○
（8 bit）	SI	0F	拡大解除（8 bit）	○	○
セレクト	DC1	11	セレクト	○	○
ディセレクト	DC3	13	ディセレクト	○	○
拡　　大	DC2	12	拡大指令（7 bit）	○	○
（7 bit）	DC4	14	拡大解除（7 bit）	○	○
水平タブ	HT	09	水平タブ移動	○	○
キャンセル	CAN	18	データのキャンセル	○	○
n 行改行	US	1F	1～15行の改行	○	○
VFU	－	－	タブ位置等の設定	○	○
印字方法	ESC, N	1B, 4E	HSパイカ	○	○
	ESC, P	1B, 50	プロポーショナル	○	○
	ESC, Q	1B, 51	コンデンス	○	○
	ESC, E	1B, 45	エリート	○	○
	ESC, H	1B, 48	HDパイカ	○	×
	ESC, K	1B, 4B	漢字	○	×
ドットスペース	ESC, SOH	1B, 01	1ドットスペース	○	○
	ESC, STX	1B, 02	2ドットスペース	○	○
	ESC, ETX	1B, 03	3ドットスペース	○	○
	ESC, EOT	1B, 04	4ドットスペース	○	○
	ESC, ENQ	1B, 05	5ドットスペース	○	○
	ESC, ACK	1B, 06	6ドットスペース	○	○
キャラクタモード	ESC, $	1B, 24	英数記号モード	○	○
	ESC, &	1B, 26	ひらがなモード	○	○
	ESC, ＃	1B, 23	内部グラフィックモード	○	×

分　　類	ニーモニック	HEXコード	機　能	PC-8821/8822	PC-8023(C)
ドット列印字モード	ESC, S	1B, 53	8 bitドット列	○	○
	ESC, I	1B, 49	16bitドット列	○	×
	ESC, V	1B, 56	8 bitドット列リピート	○	×
	ESC, W	1B, 57	16bitドット列リピート	○	×
	ESC, F	1B, 46	ドットアドレッシング	○	×
キャラクタリピート	ESC, R	1B, 52	キャラクタリピート	○	×
強調文字	ESC, !	1B, 21	強調文字セレクト	○	○
	ESC, ”	1B, 22	強調文字解除	○	○
印字モード	ESC, 〕	1B, 5D	ロジカルシークモード	○	○
	ESC, 〉	1B, 3E	片方向印字	○	×
	ESC, 〔	1B,	インクリメンタルモード	×	○
改行幅	ESC, A	1B, 41	1/6インチ改行モード	○	○
	ESC, B	1B, 42	1/8インチ改行モード	○	○
	ESC, T	1B, 54	N/120インチ改行モード	○	○
改行方向	ESC, f	1B, 66	順方向改行モード	○	○
	ESC, r	1B, 72	逆方向改行モード	○	○
水平タブ	ESC, (	1B, 28	水平タブセット	○	○
	ESC,)	1B, 29	水平タブ部分クリア	○	○
	ESC, 0	1B, 30	水平タブオールクリア	○	○
アンダーライン	ESC, X	1B, 58	アンダーライン開始	○	○
	ESC, Y	1B, 59	アンダーライン終了	○	○
レフトマージン	ESC, L	1B, 4C	印字開始位置への設定	○	○
リボン切換	ESC, C	1B, 43	リボン切替指定	○	×
外字のロード	ESC, ＊	1B, 2A	外字のロード	○	×
*ドット対応グラフィックドット数の切り換え	ESC, D	1B, 44	640ドットモード	○	×
	ESC, M	1B, 4D	960ドットモード	○	×

付録11　キャラクタ対応表

```
[ ｷｺﾞｳ ] ---------------------------------------------------------------------
      =!!  、=!"  。=!#  ，=!$  ．=!%  ・=!&  ：=!´  ；=!(  ？=!)  ！=!*
    ゛=!+  ゜=!,  ´=!-  ｀=!.  ¨=!/  ＾=!0  ￣=!1  ＿=!2  ヽ=!3  ヾ=!4
    ゝ=!5  ゞ=!6  〃=!7  仝=!8  々=!9  〆=!:  〇=!;  ー=!<  ―=!=  ‐=!>
    ／=!?  ＼=!@  ～=!A  ‖=!B  ｜=!C  …=!D  ‥=!E  ‘=!F  ’=!G  “=!H
    ”=!I  （=!J  ）=!K  〔=!L  〕=!M  ［=!N  ］=!O  ｛=!P  ｝=!Q  〈=!R
    〉=!S  《=!T  》=!U  「=!V  」=!W  『=!X  』=!Y  【=!Z  】=![  ＋=!¥
    －=!]  ±=!^  ×=!_  ÷=!⊥  ＝=!a  ≠=!b  ＜=!c  ＞=!d  ≦=!e  ≧=!f
    ∞=!g  ∴=!h  ♂=!i  ♀=!j  °=!k  ′=!l  ″=!m  ℃=!n  ￥=!o  ＄=!p
    ¢=!q  £=!r  ％=!s  ＃=!t  ＆=!u  ＊=!v  ＠=!w  §=!x  ☆=!y  ★=!z
    ○=!{  ●=!¦  ◎=!}  ◇=!~  ◆="!  □=""  ■="#  △="$  ▲="%  ▽="&
    ▼="´  ※="(  〒=")  →="*  ←="+  ↑=",  ↓="-  〓=".   ="/   ="0
     ="1   ="2   ="3   ="4   ="5   ="6   ="7   ="8   ="9   =":
     =";   ="<   =""=  =">   ="?   ="@   ="A   ="B   ="C   ="D
     ="E   ="F   ="G   ="H   ="I   ="J   ="K   ="L   ="M   ="N
     ="O   ="P   ="Q   ="R   ="S   ="T   ="U   ="V   ="W   ="X
     ="Y   ="Z   ="[   ="¥   ="]   ="^   ="_   ="⊥   ="a   ="b
     ="c   ="d   ="e   ="f   ="g   ="h   ="i   ="j   ="k   ="l
     ="m   ="n   ="o   ="p   ="q   ="r   ="s   ="t   ="u   ="v
     ="w   ="x   ="y   ="z   ="{   ="¦   ="}   ="~   =#!   =#"
     =##   =#$   =#%   =#&   =#´   =#(   =#)   =#*   =#+   =#,
     =#-   =#.   =#/  ０=#0  １=#1  ２=#2  ３=#3  ４=#4  ５=#5  ６=#6
    ７=#7  ８=#8  ９=#9   =#:   =#;   =#<   =#=   =#>   =#?   =#@
    Ａ=#A  Ｂ=#B  Ｃ=#C  Ｄ=#D  Ｅ=#E  Ｆ=#F  Ｇ=#G  Ｈ=#H  Ｉ=#I  Ｊ=#J
    Ｋ=#K  Ｌ=#L  Ｍ=#M  Ｎ=#N  Ｏ=#O  Ｐ=#P  Ｑ=#Q  Ｒ=#R  Ｓ=#S  Ｔ=#T
    Ｕ=#U  Ｖ=#V  Ｗ=#W  Ｘ=#X  Ｙ=#Y  Ｚ=#Z   =#[   =#¥   =#]   =#^
     =#_   =#⊥  ａ=#a  ｂ=#b  ｃ=#c  ｄ=#d  ｅ=#e  ｆ=#f  ｇ=#g  ｈ=#h
    ｉ=#i  ｊ=#j  ｋ=#k  ｌ=#l  ｍ=#m  ｎ=#n  ｏ=#o  ｐ=#p  ｑ=#q  ｒ=#r
    ｓ=#s  ｔ=#t  ｕ=#u  ｖ=#v  ｗ=#w  ｘ=#x  ｙ=#y  ｚ=#z   =#{   =#¦
     =#}   =#~  ぁ=$!  あ=$"  ぃ=$#  い=$$  ぅ=$%  う=$&  ぇ=$´  え=$(
    ぉ=$)  お=$*  か=$+  が=$,  き=$-  ぎ=$.  く=$/  ぐ=$0  け=$1  げ=$2
    こ=$3  ご=$4  さ=$5  ざ=$6  し=$7  じ=$8  す=$9  ず=$:  せ=$;  ぜ=$<
    そ=$=  ぞ=$>  た=$?  だ=$@  ち=$A  ぢ=$B  っ=$C  つ=$D  づ=$E  て=$F
    で=$G  と=$H  ど=$I  な=$J  に=$K  ぬ=$L  ね=$M  の=$N  は=$O  ば=$P
    ぱ=$Q  ひ=$R  び=$S  ぴ=$T  ふ=$U  ぶ=$V  ぷ=$W  へ=$X  べ=$Y  ぺ=$Z
    ほ=$[  ぼ=$¥  ぽ=$]  ま=$^  み=$_  む=$⊥  め=$a  も=$b  ゃ=$c  や=$d
    ゅ=$e  ゆ=$f  ょ=$g  よ=$h  ら=$i  り=$j  る=$k  れ=$l  ろ=$m  ゎ=$n
    わ=$o  ゐ=$p  ゑ=$q  を=$r  ん=$s   =$t   =$u   =$v   =$w   =$x
     =$y   =$z   =${   =$¦   =$}   =$~  ァ=%!  ア=%"  ィ=%#  イ=%$
    ゥ=%%  ウ=%&  ェ=%´  エ=%(  ォ=%)  オ=%*  カ=%+  ガ=%,  キ=%-  ギ=%.
    ク=%/  グ=%0  ケ=%1  ゲ=%2  コ=%3  ゴ=%4  サ=%5  ザ=%6  シ=%7  ジ=%8
    ス=%9  ズ=%:  セ=%;  ゼ=%<  ソ=%=  ゾ=%>  タ=%?  ダ=%@  チ=%A  ヂ=%B
    ッ=%C  ツ=%D  ヅ=%E  テ=%F  デ=%G  ト=%H  ド=%I  ナ=%J  ニ=%K  ヌ=%L
    ネ=%M  ノ=%N  ハ=%O  バ=%P  パ=%Q  ヒ=%R  ビ=%S  ピ=%T  フ=%U  ブ=%V
    プ=%W  ヘ=%X  ベ=%Y  ペ=%Z  ホ=%[  ボ=%¥  ポ=%]  マ=%^  ミ=%_  ム=%⊥
    メ=%a  モ=%b  ャ=%c  ヤ=%d  ュ=%e  ユ=%f  ョ=%g  ヨ=%h  ラ=%i  リ=%j
    ル=%k  レ=%l  ロ=%m  ヮ=%n  ワ=%o  ヰ=%p  ヱ=%q  ヲ=%r  ン=%s  ヴ=%t
    ヵ=%u  ヶ=%v   =%w   =%x   =%y   =%z   =%{   =%¦   =%}   =%~
    Α=&!  Β=&"  Γ=&#  Δ=&$  Ε=&%  Ζ=&&  Η=&´  Θ=&(  Ι=&)  Κ=&*
    Λ=&+  Μ=&,  Ν=&-  Ξ=&.  Ο=&/  Π=&0  Ρ=&1  Σ=&2  Τ=&3  Υ=&4
    Φ=&5  Χ=&6  Ψ=&7  Ω=&8   =&9   =&:   =&;   =&<   =&=   =&>
     =&?   =&@  α=&A  β=&B  γ=&C  δ=&D  ε=&E  ζ=&F  η=&G  θ=&H
    ι=&I  κ=&J  λ=&K  μ=&L  ν=&M  ξ=&N  ο=&O  π=&P  ρ=&Q  σ=&R
```

```
τ=&S  υ=&T  φ=&U  χ=&V  ψ=&W  ω=&X   =&Y   =&Z   =&[   =&¥
 =&]   =&^   =&_   =&⊥   =&a   =&b   =&c   =&d   =&e   =&f
 =&g   =&h   =&i   =&j   =&k   =&l   =&m   =&n   =&o   =&p
 =&q   =&r   =&s   =&t   =&u   =&v   =&w   =&x   =&y   =&z
 =&{   =&¦   =&}   =&~  А=´!  Б=´"  В=´#  Г=´$  Д=´%  Е=´&
Ё=´´  Ж=´(  З=´)  И=´*  Й=´+  К=´,  Л=´-  М=´.  Н=´/  О=´0
П=´1  Р=´2  С=´3  Т=´4  У=´5  Ф=´6  Х=´7  Ц=´8  Ч=´9  Ш=´:
Щ=´;  Ъ=´<  Ы=´=  Ь=´>  Э=´?  Ю=´@  Я=´A   =´B   =´C   =´D
 =´E   =´F   =´G   =´H   =´I   =´J   =´K   =´L   =´M   =´N
 =´O   =´P  а=´Q  б=´R  в=´S  г=´T  д=´U  е=´V  ё=´W  ж=´X
з=´Y  и=´Z  й=´[  к=´¥  л=´]  м=´^  н=´_  о=´⊥  п=´a  р=´b
с=´c  т=´d  у=´e  ф=´f  х=´g  ц=´h  ч=´i  ш=´j  щ=´k  ъ=´l
ы=´m  ь=´n  э=´o  ю=´p

[ア] ------------------------------------------------------------
亜=0! 唖=0" 娃=0# 阿=0$ 哀=0% 愛=0& 挨=0´ 姶=0( 逢=0) 葵=0*
茜=0+ 穐=0, 悪=0- 握=0. 渥=0/ 旭=00 葦=01 芦=02 鯵=03 梓=04
圧=05 斡=06 扱=07 宛=08 姐=09 虻=0: 飴=0; 絢=0< 綾=0= 鮎=0>
或=0? 粟=0@ 袷=0A 安=0B 庵=0C 按=0D 暗=0E 案=0F 闇=0G 鞍=0H
杏=0I
[イ] ------------------------------------------------------------
以=0J 伊=0K 位=0L 依=0M 偉=0N 囲=0O 夷=0P 委=0Q 威=0R 尉=0S
惟=0T 意=0U 慰=0V 易=0W 椅=0X 為=0Y 畏=0Z 異=0[ 移=0¥ 維=0]
緯=0^ 胃=0_ 萎=0⊥ 衣=0a 謂=0b 違=0c 遺=0d 医=0e 井=0f 亥=0g
域=0h 育=0i 郁=0j 磯=0k 一=0l 壱=0m 溢=0n 逸=0o 稲=0p 茨=0q
芋=0r 鰯=0s 允=0t 印=0u 咽=0v 員=0w 因=0x 姻=0y 引=0z 飲=0{
淫=0¦ 胤=0} 蔭=0~ 院=1! 陰=1" 隠=1# 韻=1$ 吋=1%
[ウ] ------------------------------------------------------------
右=1& 宇=1´ 烏=1( 羽=1) 迂=1* 雨=1+ 卯=1, 鵜=1- 窺=1. 丑=1/
碓=10 臼=11 渦=12 嘘=13 唄=14 欝=15 蔚=16 鰻=17 姥=18 厩=19
浦=1: 瓜=1; 閏=1< 噂=1= 云=1> 運=1? 雲=1@
[エ] ------------------------------------------------------------
荏=1A 餌=1B 叡=1C 営=1D 嬰=1E 影=1F 映=1G 曳=1H 栄=1I 永=1J
泳=1K 洩=1L 瑛=1M 盈=1N 穎=1O 頴=1P 英=1Q 衛=1R 詠=1S 鋭=1T
液=1U 疫=1V 益=1W 駅=1X 悦=1Y 謁=1Z 越=1[ 閲=1¥ 榎=1] 厭=1^
円=1_ 園=1⊥ 堰=1a 奄=1b 宴=1c 延=1d 怨=1e 掩=1f 援=1g 沿=1h
演=1i 炎=1j 焔=1k 煙=1l 燕=1m 猿=1n 縁=1o 艶=1p 苑=1q 薗=1r
遠=1s 鉛=1t 鴛=1u 塩=1v
[オ] ------------------------------------------------------------
於=1w 汚=1x 甥=1y 凹=1z 央=1{ 奥=1¦ 往=1} 応=1~ 押=2! 旺=2"
横=2# 欧=2$ 殴=2% 王=2& 翁=2´ 襖=2( 鴬=2) 鴎=2* 黄=2+ 岡=2,
沖=2- 荻=2. 億=2/ 屋=20 憶=21 臆=22 桶=23 牡=24 乙=25 俺=26
卸=27 恩=28 温=29 穏=2: 音=2;
[カ] ------------------------------------------------------------
下=2< 化=2= 仮=2> 何=2? 伽=2@ 価=2A 佳=2B 加=2C 可=2D 嘉=2E
夏=2F 嫁=2G 家=2H 寡=2I 科=2J 暇=2K 果=2L 架=2M 歌=2N 河=2O
火=2P 珂=2Q 禍=2R 禾=2S 稼=2T 箇=2U 花=2V 苛=2W 茄=2X 荷=2Y
華=2Z 菓=2[ 蝦=2¥ 課=2] 嘩=2^ 貨=2_ 迦=2⊥ 過=2a 霞=2b 蚊=2c
俄=2d 峨=2e 我=2f 牙=2g 画=2h 臥=2i 芽=2j 蛾=2k 賀=2l 雅=2m
餓=2n 駕=2o 介=2p 会=2q 解=2r 回=2s 塊=2t 壊=2u 廻=2v 快=2w
怪=2x 悔=2y 恢=2z 懐=2{ 戒=2¦ 拐=2} 改=2~ 魁=3! 晦=3" 械=3#
海=3$ 灰=3% 界=3& 皆=3´ 絵=3( 芥=3) 蟹=3* 開=3+ 階=3, 貝=3-
凱=3. 劾=3/ 外=30 咳=31 害=32 崖=33 慨=34 概=35 涯=36 碍=37
蓋=38 街=39 該=3: 鎧=3; 骸=3< 浬=3= 馨=3> 蛙=3? 垣=3@ 柿=3A
蛎=3B 鈎=3C 劃=3D 嚇=3E 各=3F 廓=3G 拡=3H 撹=3I 格=3J 核=3K
殻=3L 獲=3M 確=3N 穫=3O 覚=3P 角=3Q 赫=3R 較=3S 郭=3T 閣=3U
```

隔=3V 革=3W 学=3X 岳=3Y 楽=3Z 額=3[顎=3¥ 掛=3] 笠=3^ 樫=3_
橿=3` 梶=3a 鰍=3b 潟=3c 割=3d 喝=3e 恰=3f 括=3g 活=3h 渇=3i
滑=3j 葛=3k 褐=3l 轄=3m 且=3n 鰹=3o 叶=3p 椛=3q 樺=3r 鞄=3s
株=3t 兜=3u 竃=3v 蒲=3w 釜=3x 鎌=3y 噛=3z 鴨=3{ 栢=3¦ 茅=3}
萱=3~ 粥=4! 刈=4" 苅=4# 瓦=4$ 乾=4% 侃=4& 冠=4' 寒=4(刊=4)
勘=4* 勧=4+ 巻=4, 喚=4- 堪=4. 姦=4/ 完=40 官=41 寛=42 干=43
幹=44 患=45 感=46 慣=47 憾=48 換=49 敢=4: 柑=4; 桓=4< 棺=4=
款=4> 歓=4? 汗=4@ 漢=4A 澗=4B 潅=4C 環=4D 甘=4E 監=4F 看=4G
竿=4H 管=4I 簡=4J 緩=4K 缶=4L 翰=4M 肝=4N 艦=4O 莞=4P 観=4Q
諌=4R 貫=4S 還=4T 鑑=4U 間=4V 閑=4W 関=4X 陥=4Y 韓=4Z 館=4[
舘=4¥ 丸=4] 含=4^ 岸=4_ 巌=4` 玩=4a 癌=4b 眼=4c 岩=4d 翫=4e
贋=4f 雁=4g 頑=4h 顔=4i 願=4j

[キ] --

企=4k 伎=4l 危=4m 喜=4n 器=4o 基=4p 奇=4q 嬉=4r 寄=4s 岐=4t
希=4u 幾=4v 忌=4w 揮=4x 机=4y 旗=4z 既=4{ 期=4¦ 棋=4} 棄=4~
機=5! 帰=5" 毅=5# 気=5$ 汽=5% 畿=5& 祈=5' 季=5(稀=5) 紀=5*
徽=5+ 規=5, 記=5- 貴=5. 起=5/ 軌=50 輝=51 飢=52 騎=53 鬼=54
亀=55 偽=56 儀=57 妓=58 宜=59 戯=5: 技=5; 擬=5< 欺=5= 犠=5>
疑=5? 祇=5@ 義=5A 蟻=5B 誼=5C 議=5D 掬=5E 菊=5F 鞠=5G 吉=5H
吃=5I 喫=5J 桔=5K 橘=5L 詰=5M 砧=5N 杵=5O 黍=5P 却=5Q 客=5R
脚=5S 虐=5T 逆=5U 丘=5V 久=5W 仇=5X 休=5Y 及=5Z 吸=5[宮=5¥
弓=5] 急=5^ 救=5_ 朽=5` 求=5a 汲=5b 泣=5c 灸=5d 球=5e 究=5f
窮=5g 笈=5h 級=5i 糾=5j 給=5k 旧=5l 牛=5m 去=5n 居=5o 巨=5p
拒=5q 拠=5r 挙=5s 渠=5t 虚=5u 許=5v 距=5w 鋸=5x 漁=5y 禦=5z
魚=5{ 亨=5¦ 享=5} 京=5~ 供=6! 侠=6" 僑=6# 兇=6$ 競=6% 共=6&
凶=6' 協=6(匡=6) 卿=6* 叫=6+ 喬=6, 境=6- 峡=6. 強=6/ 彊=60
怯=61 恐=62 恭=63 挟=64 教=65 橋=66 況=67 狂=68 狭=69 矯=6:
胸=6; 脅=6< 興=6= 蕎=6> 郷=6? 鏡=6@ 響=6A 饗=6B 驚=6C 仰=6D
凝=6E 尭=6F 暁=6G 業=6H 局=6I 曲=6J 極=6K 玉=6L 桐=6M 粁=6N
僅=6O 勤=6P 均=6Q 巾=6R 錦=6S 斤=6T 欣=6U 欽=6V 琴=6W 禁=6X
禽=6Y 筋=6Z 緊=6[芹=6¥ 菌=6] 衿=6^ 襟=6_ 謹=6` 近=6a 金=6b
吟=6c 銀=6d

[ク] --

九=6e 倶=6f 句=6g 区=6h 狗=6i 玖=6j 矩=6k 苦=6l 躯=6m 駆=6n
駈=6o 駒=6p 具=6q 愚=6r 虞=6s 喰=6t 空=6u 偶=6v 寓=6w 遇=6x
隅=6y 串=6z 櫛=6{ 釧=6¦ 屑=6} 屈=6~ 掘=7! 窟=7" 沓=7# 靴=7$
轡=7% 窪=7& 熊=7' 隈=7(粂=7) 栗=7* 繰=7+ 桑=7, 鍬=7- 勲=7.
君=7/ 薫=70 訓=71 群=72 軍=73 郡=74

[ケ] --

卦=75 袈=76 祁=77 係=78 傾=79 刑=7: 兄=7; 啓=7< 圭=7= 珪=7>
型=7? 契=7@ 形=7A 径=7B 恵=7C 慶=7D 慧=7E 憩=7F 掲=7G 携=7H
敬=7I 景=7J 桂=7K 渓=7L 畦=7M 稽=7N 系=7O 経=7P 継=7Q 繋=7R
罫=7S 茎=7T 荊=7U 蛍=7V 計=7W 詣=7X 警=7Y 軽=7Z 頚=7[鶏=7¥
芸=7] 迎=7^ 鯨=7_ 劇=7` 戟=7a 撃=7b 激=7c 隙=7d 桁=7e 傑=7f
欠=7g 決=7h 潔=7i 穴=7j 結=7k 血=7l 訣=7m 月=7n 件=7o 倹=7p
倦=7q 健=7r 兼=7s 券=7t 剣=7u 喧=7v 圏=7w 堅=7x 嫌=7y 建=7z
憲=7{ 懸=7¦ 拳=7} 捲=7~ 検=8! 権=8" 牽=8# 犬=8$ 献=8% 研=8&
硯=8' 絹=8(県=8) 肩=8* 見=8+ 謙=8, 賢=8- 軒=8. 遣=8/ 鍵=80
険=81 顕=82 験=83 鹸=84 元=85 原=86 厳=87 幻=88 弦=89 減=8:
源=8; 玄=8< 現=8= 絃=8> 舷=8? 言=8@ 諺=8A 限=8B

[コ] --

乎=8C 個=8D 古=8E 呼=8F 固=8G 姑=8H 孤=8I 己=8J 庫=8K 弧=8L
戸=8M 故=8N 枯=8O 湖=8P 狐=8Q 糊=8R 袴=8S 股=8T 胡=8U 菰=8V
虎=8W 誇=8X 跨=8Y 鈷=8Z 雇=8[顧=8¥ 鼓=8] 五=8^ 互=8_ 伍=8`
午=8a 呉=8b 吾=8c 娯=8d 後=8e 御=8f 悟=8g 梧=8h 檎=8i 瑚=8j
碁=8k 語=8l 誤=8m 護=8n 醐=8o 乞=8p 鯉=8q 交=8r 佼=8s 侯=8t
候=8u 倖=8v 光=8w 公=8x 功=8y 効=8z 勾=8{ 厚=8¦ 口=8} 向=8~

后=9! 喉=9" 坑=9# 垢=9$ 好=9% 孔=9& 孝=9´ 宏=9(工=9) 巧=9*
巷=9+ 幸=9, 広=9- 庚=9. 康=9/ 弘=90 恒=91 慌=92 抗=93 拘=94
控=95 攻=96 昂=97 晃=98 更=99 杭=9: 校=9; 梗=9< 構=9= 江=9>
洪=9? 浩=9@ 港=9A 溝=9B 甲=9C 皇=9D 硬=9E 稿=9F 糠=9G 紅=9H
紘=9I 絞=9J 綱=9K 耕=9L 考=9M 肯=9N 肱=9O 腔=9P 膏=9Q 航=9R
荒=9S 行=9T 衡=9U 講=9V 貢=9W 購=9X 郊=9Y 酵=9Z 鉱=9[砿=9¥
鋼=9] 閤=9^ 降=9_ 項=9┴ 香=9a 高=9b 鴻=9c 剛=9d 劫=9e 号=9f
合=9g 壕=9h 拷=9i 濠=9j 豪=9k 轟=9l 麹=9m 克=9n 刻=9o 告=9p
国=9q 穀=9r 酷=9s 鵠=9t 黒=9u 獄=9v 漉=9w 腰=9x 甑=9y 忽=9z
惚=9{ 骨=9¦ 狛=9} 込=9~ 此=:! 頃=:" 今=:# 困=:$ 坤=:% 墾=:&
婚=:´ 恨=:(懇=:) 昏=:* 昆=:+ 根=:, 梱=:- 混=:. 痕=:/ 紺=:0
艮=:1 魂=:2

[サ] --

些=:3 佐=:4 叉=:5 唆=:6 嵯=:7 左=:8 差=:9 査=:: 沙=:; 瑳=:<
砂=:= 詐=:> 鎖=:? 裟=:@ 坐=:A 座=:B 挫=:C 債=:D 催=:E 再=:F
最=:G 哉=:H 塞=:I 妻=:J 宰=:K 彩=:L 才=:M 採=:N 栽=:O 歳=:P
済=:Q 災=:R 采=:S 犀=:T 砕=:U 砦=:V 祭=:W 斎=:X 細=:Y 菜=:Z
裁=:[載=:¥ 際=:] 剤=:^ 在=:_ 材=:┴ 罪=:a 財=:b 冴=:c 坂=:d
阪=:e 堺=:f 榊=:g 肴=:h 咲=:i 崎=:j 埼=:k 碕=:l 鷺=:m 作=:n
削=:o 咋=:p 搾=:q 昨=:r 朔=:s 柵=:t 窄=:u 策=:v 索=:w 錯=:x
桜=:y 鮭=:z 笹=:{ 匙=:¦ 冊=:} 刷=:~ 察=;! 拶=;" 撮=;# 擦=;$
札=;% 殺=;& 薩=;´ 雑=;(皐=;) 鯖=;* 捌=;+ 錆=;, 鮫=;- 皿=;.
晒=;/ 三=;0 傘=;1 参=;2 山=;3 惨=;4 撒=;5 散=;6 桟=;7 燦=;8
珊=;9 産=;: 算=;; 纂=;< 蚕=;= 讃=;> 賛=;? 酸=;@ 餐=;A 斬=;B
暫=;C 残=;D

[シ] --

仕=;E 仔=;F 伺=;G 使=;H 刺=;I 司=;J 史=;K 嗣=;L 四=;M 士=;N
始=;O 姉=;P 姿=;Q 子=;R 屍=;S 市=;T 師=;U 志=;V 思=;W 指=;X
支=;Y 孜=;Z 斯=;[施=;¥ 旨=;] 枝=;^ 止=;_ 死=;┴ 氏=;a 獅=;b
祉=;c 私=;d 糸=;e 紙=;f 紫=;g 肢=;h 脂=;i 至=;j 視=;k 詞=;l
詩=;m 試=;n 誌=;o 諮=;p 資=;q 賜=;r 雌=;s 飼=;t 歯=;u 事=;v
似=;w 侍=;x 児=;y 字=;z 寺=;{ 慈=;¦ 持=;} 時=;~ 次=<! 滋=<"
治=<# 爾=<$ 璽=<% 痔=<& 磁=<´ 示=<(而=<) 耳=<* 自=<+ 蒔=<,
辞=<- 汐=<. 鹿=</ 式=<0 識=<1 鴫=<2 竺=<3 軸=<4 宍=<5 雫=<6
七=<7 叱=<8 執=<9 失=<: 嫉=<; 室=<< 悉=<= 湿=<> 漆=<? 疾=<@
質=<A 実=<B 蔀=<C 篠=<D 偲=<E 柴=<F 芝=<G 屡=<H 蕊=<I 縞=<J
舎=<K 写=<L 射=<M 捨=<N 赦=<O 斜=<P 煮=<Q 社=<R 紗=<S 者=<T
謝=<U 車=<V 遮=<W 蛇=<X 邪=<Y 借=<Z 勺=<[尺=<¥ 杓=<] 灼=<^
爵=<_ 酌=<┴ 釈=<a 錫=<b 若=<c 寂=<d 弱=<e 惹=<f 主=<g 取=<h
守=<i 手=<j 朱=<k 殊=<l 狩=<m 珠=<n 種=<o 腫=<p 趣=<q 酒=<r
首=<s 儒=<t 受=<u 呪=<v 寿=<w 授=<x 樹=<y 綬=<z 需=<{ 囚=<¦
収=<} 周=<~ 宗==! 就==" 州==# 修==$ 愁==% 拾==& 洲==´ 秀==(
秋==) 終==* 繍==+ 習==, 臭==- 舟==. 蒐==/ 衆==0 襲==1 讐==2
蹴==3 輯==4 週==5 酋==6 酬==7 集==8 醜==9 什==: 住==; 充==<
十=== 従==> 戎==? 柔==@ 汁==A 渋==B 獣==C 縦==D 重==E 銃==F
叔==G 夙==H 宿==I 淑==J 祝==K 縮==L 粛==M 塾==N 熟==O 出==P
術==Q 述==R 俊==S 峻==T 春==U 瞬==V 竣==W 舜==X 駿==Y 准==Z
循==[旬==¥ 楯==] 殉==^ 淳==_ 準==┴ 潤==a 盾==b 純==c 巡==d
遵==e 醇==f 順==g 処==h 初==i 所==j 暑==k 曙==l 渚==m 庶==n
緒==o 署==p 書==q 薯==r 藷==s 諸==t 助==u 叙==v 女==w 序==x
徐==y 恕==z 鋤=={ 除==¦ 傷==} 償==~ 勝=>! 匠=>" 升=># 召=>$
哨=>% 商=>& 唱=>´ 嘗=>(奨=>) 妾=>* 娼=>+ 宵=>, 将=>- 小=>.
少=>/ 尚=>0 庄=>1 床=>2 廠=>3 彰=>4 承=>5 抄=>6 招=>7 掌=>8
捷=>9 昇=>: 昌=>; 昭=>< 晶=>= 松=>> 梢=>? 樟=>@ 樵=>A 沼=>B
消=>C 渉=>D 湘=>E 焼=>F 焦=>G 照=>H 症=>I 省=>J 硝=>K 礁=>L
祥=>M 称=>N 章=>O 笑=>P 粧=>Q 紹=>R 肖=>S 菖=>T 蒋=>U 蕉=>V
衝=>W 裳=>X 訟=>Y 証=>Z 詔=>[詳=>¥ 象=>] 賞=>^ 醤=>_ 鉦=>┴

鍾=>a 鐘=>b 障=>c 鞘=>d 上=>e 丈=>f 丞=>g 乗=>h 冗=>i 剰=>j
城=>k 場=>l 壌=>m 嬢=>n 常=>o 情=>p 擾=>q 条=>r 杖=>s 浄=>t
状=>u 畳=>v 穣=>w 蒸=>x 譲=>y 醸=>z 錠=>{ 嘱=>¦ 埴=>} 飾=>~
拭=?! 植=?" 殖=?# 燭=?$ 織=?% 職=?& 色=?´ 触=?(食=?) 蝕=?*
辱=?+ 尻=?, 伸=?- 信=?. 侵=?/ 唇=?0 娠=?1 寝=?2 審=?3 心=?4
慎=?5 振=?6 新=?7 晋=?8 森=?9 榛=?: 浸=?; 深=?< 申=?= 疹=?>
真=?? 神=?@ 秦=?A 紳=?B 臣=?C 芯=?D 薪=?E 親=?F 診=?G 身=?H
辛=?I 進=?J 針=?K 震=?L 人=?M 仁=?N 刃=?O 塵=?P 壬=?Q 尋=?R
甚=?S 尽=?T 腎=?U 訊=?V 迅=?W 陣=?X 靭=?Y

[ス] --

笥=?Z 諏=?[須=?¥ 酢=?] 図=?^ 厨=?_ 逗=?┴ 吹=?a 垂=?b 帥=?c
推=?d 水=?e 炊=?f 睡=?g 粋=?h 翠=?i 衰=?j 遂=?k 酔=?l 錐=?m
錘=?n 随=?o 瑞=?p 髄=?q 崇=?r 嵩=?s 数=?t 枢=?u 趨=?v 雛=?w
据=?x 杉=?y 椙=?z 菅=?{ 頗=?¦ 雀=?} 裾=?~ 澄=@! 摺=@" 寸=@#

[セ] --

世=@$ 瀬=@% 畝=@& 是=@´ 凄=@(制=@) 勢=@* 姓=@+ 征=@, 性=@-
成=@. 政=@/ 整=@0 星=@1 晴=@2 棲=@3 栖=@4 正=@5 清=@6 牲=@7
生=@8 盛=@9 精=@: 聖=@; 声=@< 製=@= 西=@> 誠=@? 誓=@@ 請=@A
逝=@B 醒=@C 青=@D 静=@E 斉=@F 税=@G 脆=@H 隻=@I 席=@J 惜=@K
戚=@L 斥=@M 昔=@N 析=@O 石=@P 積=@Q 籍=@R 績=@S 脊=@T 責=@U
赤=@V 跡=@W 蹟=@X 碩=@Y 切=@Z 拙=@[接=@¥ 摂=@] 折=@^ 設=@_
窃=@┴ 節=@a 説=@b 雪=@c 絶=@d 舌=@e 蝉=@f 仙=@g 先=@h 千=@i
占=@j 宣=@k 専=@l 尖=@m 川=@n 戦=@o 扇=@p 撰=@q 栓=@r 栴=@s
泉=@t 浅=@u 洗=@v 染=@w 潜=@x 煎=@y 煽=@z 旋=@{ 穿=@¦ 箭=@}
線=@~ 繊=A! 羨=A" 腺=A# 舛=A$ 船=A% 薦=A& 詮=A´ 賎=A(践=A)
選=A* 遷=A+ 銭=A, 銑=A- 閃=A. 鮮=A/ 前=A0 善=A1 漸=A2 然=A3
全=A4 禅=A5 繕=A6 膳=A7 糎=A8

[ソ] --

噌=A9 塑=A: 岨=A; 措=A< 曾=A= 曽=A> 楚=A? 狙=A@ 疏=AA 疎=AB
礎=AC 祖=AD 租=AE 粗=AF 素=AG 組=AH 蘇=AI 訴=AJ 阻=AK 遡=AL
鼠=AM 僧=AN 創=AO 双=AP 叢=AQ 倉=AR 喪=AS 壮=AT 奏=AU 爽=AV
宋=AW 層=AX 匝=AY 惣=AZ 想=A[捜=A¥ 掃=A] 挿=A^ 掻=A_ 操=A┴
早=Aa 曹=Ab 巣=Ac 槍=Ad 槽=Ae 漕=Af 燥=Ag 争=Ah 痩=Ai 相=Aj
窓=Ak 糟=Al 総=Am 綜=An 聡=Ao 草=Ap 荘=Aq 葬=Ar 蒼=As 藻=At
装=Au 走=Av 送=Aw 遭=Ax 鎗=Ay 霜=Az 騒=A{ 像=A¦ 増=A} 憎=A~
臓=B! 蔵=B" 贈=B# 造=B$ 促=B% 側=B& 則=B´ 即=B(息=B) 捉=B*
束=B+ 測=B, 足=B- 速=B. 俗=B/ 属=B0 賊=B1 族=B2 続=B3 卒=B4
袖=B5 其=B6 揃=B7 存=B8 孫=B9 尊=B: 損=B; 村=B< 遜=B=

[タ] --

他=B> 多=B? 太=B@ 汰=BA 詑=BB 唾=BC 堕=BD 妥=BE 惰=BF 打=BG
柁=BH 舵=BI 楕=BJ 陀=BK 駄=BL 騨=BM 体=BN 堆=BO 対=BP 耐=BQ
岱=BR 帯=BS 待=BT 怠=BU 態=BV 戴=BW 替=BX 泰=BY 滞=BZ 胎=B[
腿=B¥ 苔=B] 袋=B^ 貸=B_ 退=B┴ 逮=Ba 隊=Bb 黛=Bc 鯛=Bd 代=Be
台=Bf 大=Bg 第=Bh 醍=Bi 題=Bj 鷹=Bk 滝=Bl 瀧=Bm 卓=Bn 啄=Bo
宅=Bp 托=Bq 択=Br 拓=Bs 沢=Bt 濯=Bu 琢=Bv 託=Bw 鐸=Bx 濁=By
諾=Bz 茸=B{ 凧=B¦ 蛸=B} 只=B~ 叩=C! 但=C" 達=C# 辰=C$ 奪=C%
脱=C& 巽=C´ 竪=C(辿=C) 棚=C* 谷=C+ 狸=C, 鱈=C- 樽=C. 誰=C/
丹=C0 単=C1 嘆=C2 坦=C3 担=C4 探=C5 旦=C6 歎=C7 淡=C8 湛=C9
炭=C: 短=C; 端=C< 箪=C= 綻=C> 耽=C? 胆=C@ 蛋=CA 誕=CB 鍛=CC
団=CD 壇=CE 弾=CF 断=CG 暖=CH 檀=CI 段=CJ 男=CK 談=CL

[チ] --

値=CM 知=CN 地=CO 弛=CP 恥=CQ 智=CR 池=CS 痴=CT 稚=CU 置=CV
致=CW 蜘=CX 遅=CY 馳=CZ 築=C[畜=C¥ 竹=C] 筑=C^ 蓄=C_ 逐=C┴
秩=Ca 窒=Cb 茶=Cc 嫡=Cd 着=Ce 中=Cf 仲=Cg 宙=Ch 忠=Ci 抽=Cj
昼=Ck 柱=Cl 注=Cm 虫=Cn 衷=Co 註=Cp 酎=Cq 鋳=Cr 駐=Cs 樗=Ct
瀦=Cu 猪=Cv 苧=Cw 著=Cx 貯=Cy 丁=Cz 兆=C{ 凋=C¦ 喋=C} 寵=C~

```
        帖=D!  帳=D"  庁=D#  弔=D$  張=D%  彫=D&  徴=D'  懲=D(  挑=D)  暢=D*
        朝=D+  潮=D,  牒=D-  町=D.  眺=D/  聴=D0  脹=D1  腸=D2  蝶=D3  調=D4
        諜=D5  超=D6  跳=D7  銚=D8  長=D9  頂=D:  鳥=D;  勅=D<  捗=D=  直=D>
        朕=D?  沈=D@  珍=DA  賃=DB  鎮=DC  陳=DD
[ ツ ] -------------------------------------------------------------------------
        津=DE  墜=DF  椎=DG  槌=DH  追=DI  鎚=DJ  痛=DK  通=DL  塚=DM  栂=DN
        掴=DO  槻=DP  佃=DQ  漬=DR  柘=DS  辻=DT  蔦=DU  綴=DV  鍔=DW  椿=DX
        潰=DY  坪=DZ  壷=D[  嬬=D¥  紬=D]  爪=D^  吊=D_  釣=D⊥  鶴=Da
[ テ ] -------------------------------------------------------------------------
        亭=Db  低=Dc  停=Dd  偵=De  剃=Df  貞=Dg  呈=Dh  堤=Di  定=Dj  帝=Dk
        底=Dl  庭=Dm  廷=Dn  弟=Do  悌=Dp  抵=Dq  挺=Dr  提=Ds  梯=Dt  汀=Du
        碇=Dv  禎=Dw  程=Dx  締=Dy  艇=Dz  訂=D{  諦=D¦  蹄=D}  逓=D~  邸=E!
        鄭=E"  釘=E#  鼎=E$  泥=E%  摘=E&  擢=E'  敵=E(  滴=E)  的=E*  笛=E+
        適=E,  鏑=E-  溺=E.  哲=E/  徹=E0  撤=E1  轍=E2  迭=E3  鉄=E4  典=E5
        填=E6  天=E7  展=E8  店=E9  添=E:  纏=E;  甜=E<  貼=E=  転=E>  顛=E?
        点=E@  伝=EA  殿=EB  澱=EC  田=ED  電=EE
[ ト ] -------------------------------------------------------------------------
        兎=EF  吐=EG  堵=EH  塗=EI  妬=EJ  屠=EK  徒=EL  斗=EM  杜=EN  渡=EO
        登=EP  菟=EQ  賭=ER  途=ES  都=ET  鍍=EU  砥=EV  礪=EW  努=EX  度=EY
        土=EZ  奴=E[  怒=E¥  倒=E]  党=E^  冬=E_  凍=E⊥  刀=Ea  唐=Eb  塔=Ec
        塘=Ed  套=Ee  宕=Ef  島=Eg  嶋=Eh  悼=Ei  投=Ej  搭=Ek  東=El  桃=Em
        檮=En  棟=Eo  盗=Ep  淘=Eq  湯=Er  濤=Es  灯=Et  燈=Eu  当=Ev  痘=Ew
        禱=Ex  等=Ey  答=Ez  筒=E{  糖=E¦  統=E}  到=E~  董=F!  蕩=F"  藤=F#
        討=F$  謄=F%  豆=F&  踏=F'  逃=F(  透=F)  鐙=F*  陶=F+  頭=F,  騰=F-
        闘=F.  働=F/  動=F0  同=F1  堂=F2  導=F3  憧=F4  撞=F5  洞=F6  瞳=F7
        童=F8  胴=F9  萄=F:  道=F;  銅=F<  峠=F=  鴇=F>  匿=F?  得=F@  徳=FA
        瀆=FB  特=FC  督=FD  禿=FE  篤=FF  毒=FG  独=FH  読=FI  栃=FJ  橡=FK
        凸=FL  突=FM  椴=FN  届=FO  鳶=FP  苫=FQ  寅=FR  酉=FS  瀞=FT  噸=FU
        屯=FV  惇=FW  敦=FX  沌=FY  豚=FZ  遁=F[  頓=F¥  呑=F]  曇=F^  鈍=F_

[ ナ ] -------------------------------------------------------------------------
        奈=F⊥  那=Fa  内=Fb  乍=Fc  凪=Fd  薙=Fe  謎=Ff  灘=Fg  捺=Fh  鍋=Fi
        楢=Fj  馴=Fk  縄=Fl  畷=Fm  南=Fn  楠=Fo  軟=Fp  難=Fq  汝=Fr
[ ニ ] -------------------------------------------------------------------------
        二=Fs  尼=Ft  弐=Fu  邇=Fv  匂=Fw  賑=Fx  肉=Fy  虹=Fz  廿=F{  日=F¦
        乳=F}  入=F~  如=G!  尿=G"  韮=G#  任=G$  妊=G%  忍=G&  認=G'
[ ヌ ] -------------------------------------------------------------------------
        濡=G(
[ ネ ] -------------------------------------------------------------------------
        禰=G)  祢=G*  寧=G+  葱=G,  猫=G-  熱=G.  年=G/  念=G0  捻=G1  撚=G2
        燃=G3  粘=G4
[ ノ ] -------------------------------------------------------------------------
        乃=G5  廼=G6  之=G7  埜=G8  囊=G9  悩=G:  濃=G;  納=G<  能=G=  脳=G>
        膿=G?  農=G@  覗=GA  蚤=GB
[ ハ ] -------------------------------------------------------------------------
        巴=GC  把=GD  播=GE  覇=GF  杷=GG  波=GH  派=GI  琶=GJ  破=GK  婆=GL
        罵=GM  芭=GN  馬=GO  俳=GP  廃=GQ  拝=GR  排=GS  敗=GT  杯=GU  盃=GV
        牌=GW  背=GX  肺=GY  輩=GZ  配=G[  倍=G¥  培=G]  媒=G^  梅=G_  楳=G⊥
        煤=Ga  狽=Gb  買=Gc  売=Gd  賠=Ge  陪=Gf  這=Gg  蠅=Gh  秤=Gi  矧=Gj
        萩=Gk  伯=Gl  剥=Gm  博=Gn  拍=Go  柏=Gp  泊=Gq  白=Gr  箔=Gs  粕=Gt
        舶=Gu  薄=Gv  迫=Gw  曝=Gx  漠=Gy  爆=Gz  縛=G{  莫=G¦  駁=G}  麦=G~
        函=H!  箱=H"  硲=H#  箸=H$  肇=H%  筈=H&  櫨=H'  幡=H(  肌=H)  畑=H*
        畠=H+  八=H,  鉢=H-  潑=H.  発=H/  醱=H0  髪=H1  伐=H2  罰=H3  抜=H4
        筏=H5  閥=H6  鳩=H7  噺=H8  塙=H9  蛤=H:  隼=H;  伴=H<  判=H=  半=H>
        反=H?  叛=H@  帆=HA  搬=HB  斑=HC  板=HD  氾=HE  汎=HF  版=HG  犯=HH
        班=HI  畔=HJ  繁=HK  般=HL  藩=HM  販=HN  範=HO  釆=HP  煩=HQ  頒=HR
        飯=HS  挽=HT  晩=HU  番=HV  盤=HW  磐=HX  蕃=HY  蛮=HZ
```

```
[ ヒ ] -----------------------------------------------------
       匪=H[  卑=H¥  否=H]  妃=H^  庇=H_  彼=H⊥  悲=Ha  扉=Hb  批=Hc  披=Hd
       斐=He  比=Hf  泌=Hg  疲=Hh  皮=Hi  碑=Hj  秘=Hk  緋=Hl  罷=Hm  肥=Hn
       被=Ho  誹=Hp  費=Hq  避=Hr  非=Hs  飛=Ht  樋=Hu  簸=Hv  備=Hw  尾=Hx
       微=Hy  枇=Hz  毘=H{  琵=H¦  眉=H}  美=H~  鼻=I!  柊=I"  稗=I#  匹=I$
       疋=I%  髭=I&  彦=I´  膝=I(  菱=I)  肘=I*  弼=I+  必=I,  畢=I-  筆=I.
       逼=I/  桧=I0  姫=I1  媛=I2  紐=I3  百=I4  謬=I5  俵=I6  彪=I7  標=I8
       氷=I9  漂=I:  瓢=I;  票=I<  表=I=  評=I>  豹=I?  廟=I@  描=IA  病=IB
       秒=IC  苗=ID  錨=IE  鋲=IF  蒜=IG  蛭=IH  鰭=II  品=IJ  彬=IK  斌=IL
       浜=IM  瀕=IN  貧=IO  賓=IP  頻=IQ  敏=IR  瓶=IS
[ フ ] -----------------------------------------------------
       不=IT  付=IU  埠=IV  夫=IW  婦=IX  富=IY  冨=IZ  布=I[  府=I¥  怖=I]
       扶=I^  敷=I_  斧=I⊥  普=Ia  浮=Ib  父=Ic  符=Id  腐=Ie  膚=If  芙=Ig
       譜=Ih  負=Ii  賦=Ij  赴=Ik  阜=Il  附=Im  侮=In  撫=Io  武=Ip  舞=Iq
       葡=Ir  蕪=Is  部=It  封=Iu  楓=Iv  風=Iw  葺=Ix  蕗=Iy  伏=Iz  副=I{
       復=I¦  幅=I}  服=I~  福=J!  腹=J"  複=J#  覆=J$  淵=J%  弗=J&  払=J´
       沸=J(  仏=J)  物=J*  鮒=J+  分=J,  吻=J-  噴=J.  墳=J/  憤=J0  扮=J1
       焚=J2  奮=J3  粉=J4  糞=J5  紛=J6  雰=J7  文=J8  聞=J9
[ ヘ ] -----------------------------------------------------
       丙=J:  併=J;  兵=J<  塀=J=  幣=J>  平=J?  弊=J@  柄=JA  並=JB  蔽=JC
       閉=JD  陛=JE  米=JF  頁=JG  僻=JH  壁=JI  癖=JJ  碧=JK  別=JL  瞥=JM
       蔑=JN  箆=JO  偏=JP  変=JQ  片=JR  篇=JS  編=JT  辺=JU  返=JV  遍=JW
       便=JX  勉=JY  娩=JZ  弁=J[  鞭=J¥
[ ホ ] -----------------------------------------------------
       保=J]  舗=J^  鋪=J_  圃=J⊥  捕=Ja  歩=Jb  甫=Jc  補=Jd  輔=Je  穂=Jf
       募=Jg  墓=Jh  慕=Ji  戊=Jj  暮=Jk  母=Jl  簿=Jm  菩=Jn  倣=Jo  俸=Jp
       包=Jq  呆=Jr  報=Js  奉=Jt  宝=Ju  峰=Jv  峯=Jw  崩=Jx  庖=Jy  抱=Jz
       捧=J{  放=J¦  方=J}  朋=J~  法=K!  泡=K"  烹=K#  砲=K$  縫=K%  胞=K&
       芳=K´  萌=K(  蓬=K)  蜂=K*  褒=K+  訪=K,  豊=K-  邦=K.  鋒=K/  飽=K0
       鳳=K1  鵬=K2  乏=K3  亡=K4  傍=K5  剖=K6  坊=K7  妨=K8  帽=K9  忘=K:
       忙=K;  房=K<  暴=K=  望=K>  某=K?  棒=K@  冒=KA  紡=KB  肪=KC  膨=KD
       謀=KE  貌=KF  貿=KG  鉾=KH  防=KI  吠=KJ  頬=KK  北=KL  僕=KM  卜=KN
       墨=KO  撲=KP  朴=KQ  牧=KR  睦=KS  穆=KT  釦=KU  勃=KV  没=KW  殆=KX
       堀=KY  幌=KZ  奔=K[  本=K¥  翻=K]  凡=K^  盆=K_
[ マ ] -----------------------------------------------------
       摩=K⊥  磨=Ka  魔=Kb  麻=Kc  埋=Kd  妹=Ke  昧=Kf  枚=Kg  毎=Kh  哩=Ki
       槙=Kj  幕=Kk  膜=Kl  枕=Km  鮪=Kn  柾=Ko  鱒=Kp  桝=Kq  亦=Kr  俣=Ks
       又=Kt  抹=Ku  末=Kv  沫=Kw  迄=Kx  侭=Ky  繭=Kz  麿=K{  万=K¦  慢=K}
       満=K~  漫=L!  蔓=L"
[ ミ ] -----------------------------------------------------
       味=L#  未=L$  魅=L%  巳=L&  箕=L´  岬=L(  密=L)  蜜=L*  湊=L+  蓑=L,
       稔=L-  脈=L.  妙=L/  粍=L0  民=L1  眠=L2
[ ム ] -----------------------------------------------------
       務=L3  夢=L4  無=L5  牟=L6  矛=L7  霧=L8  鵡=L9  椋=L:  婿=L;  娘=L<

[ メ ] -----------------------------------------------------
       冥=L=  名=L>  命=L?  明=L@  盟=LA  迷=LB  銘=LC  鳴=LD  姪=LE  牝=LF
       滅=LG  免=LH  棉=LI  綿=LJ  緬=LK  面=LL  麺=LM
[ モ ] -----------------------------------------------------
       摸=LN  模=LO  茂=LP  妄=LQ  孟=LR  毛=LS  猛=LT  盲=LU  網=LV  耗=LW
       蒙=LX  儲=LY  木=LZ  黙=L[  目=L¥  杢=L]  勿=L^  餅=L_  尤=L⊥  戻=La
       籾=Lb  貰=Lc  問=Ld  悶=Le  紋=Lf  門=Lg  匁=Lh
[ ヤ ] -----------------------------------------------------
       也=Li  冶=Lj  夜=Lk  爺=Ll  耶=Lm  野=Ln  弥=Lo  矢=Lp  厄=Lq  役=Lr
       約=Ls  薬=Lt  訳=Lu  躍=Lv  靖=Lw  柳=Lx  薮=Ly  鑓=Lz
[ ユ ] -----------------------------------------------------
       愉=L{  愈=L¦  油=L}  癒=L~  諭=M!  輸=M"  唯=M#  佑=M$  優=M%  勇=M&
```

友=M´ 宥=M(幽=M) 悠=M* 憂=M+ 揖=M, 有=M- 柚=M. 湧=M/ 涌=M0
猶=M1 猷=M2 由=M3 祐=M4 裕=M5 誘=M6 遊=M7 邑=M8 郵=M9 雄=M:
融=M; 夕=M<

[ヨ] --

予=M= 余=M> 与=M? 誉=M@ 輿=MA 預=MB 傭=MC 幼=MD 妖=ME 容=MF
庸=MG 揚=MH 揺=MI 擁=MJ 曜=MK 楊=ML 様=MM 洋=MN 溶=MO 熔=MP
用=MQ 窯=MR 羊=MS 耀=MT 葉=MU 蓉=MV 要=MW 謡=MX 踊=MY 遥=MZ
陽=M[養=M¥ 慾=M] 抑=M^ 欲=M_ 沃=M┴ 浴=Ma 翌=Mb 翼=Mc 淀=Md

[ラ] --

羅=Me 螺=Mf 裸=Mg 来=Mh 莱=Mi 頼=Mj 雷=Mk 洛=Ml 絡=Mm 落=Mn
酪=Mo 乱=Mp 卵=Mq 嵐=Mr 欄=Ms 濫=Mt 藍=Mu 蘭=Mv 覧=Mw

[リ] --

利=Mx 吏=My 履=Mz 李=M{ 梨=M¦ 理=M} 璃=M~ 痢=N! 裏=N" 裡=N#
里=N$ 離=N% 陸=N& 律=N´ 率=N(立=N) 葎=N* 掠=N+ 略=N, 劉=N-
流=N. 溜=N/ 琉=N0 留=N1 硫=N2 粒=N3 隆=N4 竜=N5 龍=N6 侶=N7
慮=N8 旅=N9 虜=N: 了=N; 亮=N< 僚=N= 両=N> 凌=N? 寮=N@ 料=NA
梁=NB 涼=NC 猟=ND 療=NE 瞭=NF 稜=NG 糧=NH 良=NI 諒=NJ 遼=NK
量=NL 陵=NM 領=NN 力=NO 緑=NP 倫=NQ 厘=NR 林=NS 淋=NT 燐=NU
琳=NV 臨=NW 輪=NX 隣=NY 鱗=NZ 麟=N[

[ル] --

瑠=N¥ 塁=N] 涙=N^ 累=N_ 類=N┴

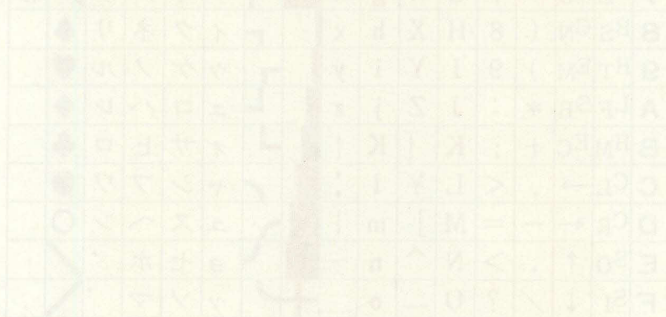
[レ] --

令=Na 伶=Nb 例=Nc 冷=Nd 励=Ne 嶺=Nf 怜=Ng 玲=Nh 礼=Ni 苓=Nj
鈴=Nk 隷=Nl 零=Nm 霊=Nn 麗=No 齢=Np 暦=Nq 歴=Nr 列=Ns 劣=Nt
烈=Nu 裂=Nv 廉=Nw 恋=Nx 憐=Ny 漣=Nz 煉=N{ 簾=N¦ 練=N} 聯=N~
蓮=O! 連=O" 錬=O#

[ロ] --

呂=O$ 魯=O% 櫓=O& 炉=O´ 賂=O(路=O) 露=O* 労=O+ 婁=O, 廊=O-
弄=O. 朗=O/ 楼=O0 榔=O1 浪=O2 漏=O3 牢=O4 狼=O5 籠=O6 老=O7
聾=O8 蝋=O9 郎=O: 六=O; 麓=O< 禄=O= 肋=O> 録=O? 論=O@

[ワ] --

倭=OA 和=OB 話=OC 歪=OD 賄=OE 脇=OF 惑=OG 枠=OH 鷲=OI 亙=OJ
亘=OK 鰐=OL 詫=OM 藁=ON 蕨=OO 椀=OP 湾=OQ 碗=OR 腕=OS

付録12　キャラクタコード表

ASCIIコード表（キャラクタセット）

上位４ビット→　／　下位４ビット↓

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		D_E		0	@	P		p				ー	タ	ミ		
1	S_H	D_1	!	1	A	Q	a	q			。	ア	チ	ム		円
2	S_X	D_2	"	2	B	R	b	r			「	イ	ツ	メ		年
3	E_X	D_3	#	3	C	S	c	s			」	ウ	テ	モ		月
4	E_T	D_4	$	4	D	T	d	t			、	エ	ト	ヤ		日
5	E_Q	N_K	%	5	E	U	e	u			・	オ	ナ	ユ		時
6	A_K	S_N	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		分
7	B_L	E_B	'	7	G	W	g	w			ァ	キ	ヌ	ラ		秒
8	B_S	C_N	(	8	H	X	h	x			ィ	ク	ネ	リ	♠	
9	H_T	E_M	)	9	I	Y	i	y			ゥ	ケ	ノ	ル	♥	
A	L_F	S_B	*	:	J	Z	j	z			ェ	コ	ハ	レ	♦	
B	H_M	E_C	+	;	K	[	K	{			ォ	サ	ヒ	ロ	♣	
C	C_L	→	,	<	L	¥	l	¦			ャ	シ	フ	ワ	●	
D	C_R	←	−	=	M	]	m	}			ュ	ス	ヘ	ン	○	
E	S_O	↑	.	>	N	^	n	~			ョ	セ	ホ	゛		
F	S_I	↓	/	?	O	_	o				ッ	ソ	マ	゜		

<コントロールコード>

16進	10進	シンボル	シンボルの意味
00	0		null
01	1	SH	Start of Heading（ヘッディング開始）
02	2	SX	Start of Text（テキスト開始）
03	3	EX	End of Text（テキスト終了）
04	4	ET	End of Transmission（伝送終了）
05	5	EQ	Enquiry（問合わせ）
06	6	AK	Acknowledge（肯定応答）
07	7	BL	Bell（ベル、ブザー）
08	8	BS	Back Space（後退）
09	9	HT	Horizontal Tabulation（水平タブ）
0A	10	LF	Line Feed（改行）
0B	11	HM	Home (VT) Vertical Tabulation（垂直タブ）
0C	12	CL	Clear (FF) Form Feed（改頁）
0D	13	CR	Carriage Return（復帰）
0E	14	SO	Sift-out（シフトアウト）
0F	15	SI	Sift-in（シフトイン）
10	16	DE	Data Link Escape（伝送制御拡張）
11	17	D1	Device Control1（装置制御1）
12	18	D2	Device Control2（装置制御2）
13	19	D3	Device Control3（装置制御3）
14	20	D4	Device Control4（装置制御4）
15	21	NK	Negative Acknowledge（否定応答）
16	22	SN	Synchronous idle（同期信号）
17	23	EB	End of Transmission Block（伝送ブロック終了）
18	24	CN	Cancel（取消し）
19	25	EM	End of Medium（媒体終端）
1A	26	SB	Substitute（文字置換）
1B	27	EC	Escape（拡張）
1C	28	→	(FS) File Separator（ファイル分離）
1D	29	←	(GS) Group Separator（グループ分離）
1E	30	↑	(RS) Record Separator（レコード分離）
1F	31	↓	(US) Unit Separator（ユニット分離）

ＥＢＣＤＩＣ（カナ入り）コード表

上位４ビット→

下位４ビット↓

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL		DS		SP	&	—			ソ						0
1			SOS				／		aア	jタ			A	J		1
2			FS						bイ	kチ	sヘ		B	K	S	2
3		TM							cウ	lツ	tホ		C	L	T	3
4	PF	RES	BYP	PN					dエ	mテ	uマ		D	M	U	4
5	HT	NL	LF	RS					eオ	nト	vミ		E	N	V	5
6	LC	BS	EOB	UC					fカ	oナ	wム		F	O	W	6
7	DL	IL	PRE	EOT					gキ	pニ	xメ		G	P	X	7
8									hク	qヌ	yモ		H	Q	Y	8
9									iケ	rネ	zヤ		I	R	Z	9
A		CC	SM		¢	!		:	コ	ノ	ユ	レ				
B					・	$	,	#				ロ				
C					<	*	%	@	サ		ヨ	ワ				
D					(	)	—	″	シ	ハ	ラ	ン				
E					+	;	>	=	ス	ヒ	リ	゛				
F					|	¬	?	"	セ	フ	ル	゜				

＜コントロールコード＞

NUL	Null	BS	Back Space	EOB	End Of Block
PF	Punch Off	IL	Idle	PRE	Prefix
HT	Horizontal Tab	CC	Cursor Control	PN	Punch On
LC	Lower Case	DS	Digit Select	RS	Reader Stop
DL	Delete	SOS	Start Significance	UC	Upper Case
TM	Tape Mark	FS	Field Separator	EOT	End Of Transmission
RES	Restore	BYP	Bypass	SM	Set Mode
NL	New Line	LF	Line Feed	SP	Space

付録13 USING文フォーマット一覧表

フォーマット	機　　能	例
!	文字列の最初の文字だけ出力	PRINT USING "F.1[!]";"abcde" F.1[a]
& ‥ &（n文字）	始めのn文字を左づめで出力	PRINT USING "F.2:[& &]";"abcde" F.2:[abcd]
@	1つの@に対して、1つの文字列を出力	PRINT USING "F.3:[@]";"abcde" F.3:[abcde]
####	数値を右づめで表示	PRINT USING "F.4:[####]";123.456 F.4:[123]
####.#	小数点の位置を指定	PRINT USING "F.5:[####.#]";123.456 F.5:[123.5]
+###.#	数値の前に符号(+，－)をつける	PRINT USING "F.6:[+###.#]";123.456 F.6:[+123.5]
###.#+	数値の後に符号(+，－)をつける	PRINT USING "F.7:[###.#+]";123.456,-123.456 F.7:[123.5+]F.7:[123.5-] PRINT USING "F.8:[###.#-]";123.456,-123.456 F.8:[123.5]F.8:[123.5-]
**####.#	空白部分を"*"で埋める	PRINT USING "F.9:[**####.#]";123.456 F.9:[***123.5]
¥¥####.#	数値の直前に"¥"をつける	PRINT USING "F.10:[¥¥####.#]";123.456 F.10:[¥123.5]
**¥##.#	**と¥¥の両方の機能となる	PRINT USING "F.11:[**¥##.#]";123.456 F.11:[*¥123.5]
#####,	3桁毎に","で区切って出力	PRINT USING "F.12:[######,]";1234.56 F.12:[1,235]
###^^^^	数値を指数形式で出力	PRINT USING "F.13:[###^^^^]";1234.56 F.13:[12E+02]
####_,####	"_"に続く1文字を単に文字として出力	PRINT USING "F.14:[####_,####]";123,123 F.14:[123, 123]

付録14　ニーモニック対応表(Intel 8080→Z－80)

＊印はPC-8801モニタで使える相対ジャンプ命令

```
> --- Intel 8080 Mnemonics to Z-80 Mnemonics --- <

      r  :  Register  ( A,B,C,D,E,H,L )
      n  :  Constant (  8 bits )
      nn :  Constant ( 16 bits )

    [ i8080 ]                    [ Z80 ]

     ACI   n                      ADC  A,n
     ADC   M                      ADC  A,(HL)
     ADC   r                      ADC  A,r
     ADD   M                      ADD  A,(HL)
     ADD   r                      ADD  A,r
     ADI   n                      ADD  A,n
     ANA   M                      AND  (HL)
     ANA   r                      AND  r
     ANI   n                      AND  n
     CALL  nn                     CALL nn
     CC    nn                     CALL C,nn
     CM    nn                     CALL M,nn
     CMA                          CPL
     CMC                          CCF
     CMP   M                      CP   (HL)
     CMP   r                      CP   r
     CNC   nn                     CALL NC,nn
     CNZ   nn                     CALL NZ,nn
     CP    nn                     CALL P,nn
     CPE   nn                     CALL PE,nn
     CPI   n                      CP   n
     CPO   nn                     CALL PO,nn
     CZ    nn                     CALL Z,nn
     DAA                          DAA
     DAD   B                      ADD  HL,BC
     DAD   D                      ADD  HL,DE
     DAD   H                      ADD  HL,HL
     DAD   SP                     ADD  HL,SP
     DCR   M                      DEC  (HL)
     DCR   r                      DEC  r
     DCX   B                      DEC  BC
     DCX   D                      DEC  DE
     DCX   H                      DEC  HL
     DCX   SP                     DEC  SP
     DI                           DI
     DJNZ  n *                    DJNZ n
     EI                           EI
     HLT                          HALT
     IN    n                      IN   A,(n)
     INR   M                      INC  (HL)
     INR   r                      INC  r
     INX   B                      INC  BC
     INX   D                      INC  DE
     INX   H                      INC  HL
```

```
INX  SP        INC  SP
JC   nn        JP   C,nn
JM   nn        JP   M,nn
JMP  nn        JP   nn
JMPR n *       JR   n
JNC  nn        JP   NC,nn
JNZ  nn        JP   NZ,nn
JP   nn        JP   P,nn
JPE  nn        JP   PE,nn
JPO  nn        JP   PO,nn
JRC  n *       JR   C,n
JRNC n *       JR   NC,n
JRNZ n *       JR   NZ,n
JRZ  n *       JR   Z,n
JZ   nn        JP   Z,nn
LDA  nn        LD   A,(nn)
LDAX B         LD   A,(BC)
LDAX D         LD   A,(DE)
LHLD nn        LD   HL,(nn)
LXI  B,nn      LD   BC,nn
LXI  D,nn      LD   DE,nn
LXI  H,nn      LD   HL,nn
LXI  SP,nn     LD   SP,nn
MOV  M,r       LD   (HL),r
MOV  r,M       LD   r,(HL)
MOV  r,r´      LD   r,r´
MVI  M,n       LD   (HL),n
MVI  r,n       LD   r,n
NOP            NOP
ORA  M         OR   (HL)
ORA  r         OR   r
ORI  n         OR   n
OUT  n         OUT  (n),A
PCHL           JP   (HL)
POP  B         POP  BC
POP  D         POP  DE
POP  H         POP  HL
POP  PSW       POP  AF
PUSH B         PUSH BC
PUSH D         PUSH DE
PUSH H         PUSH HL
PUSH PSW       PUSH AF
RAL            RLA
RAR            RRA
RC             RET  C
RET            RET
RLC            RLCA
RM             RET  M
RNC            RET  NC
RNZ            RET  NZ
RP             RET  P
RPE            RET  PE
RPO            RET  PO
RRC            RRCA
RST  0         RST  0H
RST  1         RST  8H
RST  2         RST  10H
RST  3         RST  18H
RST  4         RST  20H
RST  5         RST  28H
RST  6         RST  30H
RST  7         RST  38H
RZ             RET  Z
```

SBB M	SBC A,(HL)
SBB r	SBC A,r
SBI n	SBC A,n
SHLD nn	LD (nn),HL
SPHL	LD SP,HL
STA nn	LD (nn),A
STAX B	LD (BC),A
STAX D	LD (DE),A
STC	SCF
SUB M	SUB (HL)
SUB r	SUB r
SUI n	SUB n
XCHG	EX DE,HL
XRA M	XOR (HL)
XRA r	XOR r
XRI n	XOR n
XTHL	EX (SP),HL

付録15　PC−8801　ROM　Ver1.0　vs　Ver1.1

PC−8801のROMは現在２種類のものが出回っています。

ここでは、Ver1.0からVer1.1への主な変更点と、異なる部分のメモリダンプを示します。

○主な変更点

1. ラベル名の２重定義エラーが出た後でコマンドのタイプミスをした場合でもBASICが正常に動作するようになっています。
2. 20行モードの画面でCOPY4, COPY5を使用した場合でも問題なく印字できるようになっています。
3. 【DEL】キーの先行入力をなくし、キーを離すと同時にその動作をやめるようになっています。
4. ターミナルモードでACOSと接続した場合に画面が正しく出るようになっています。
5. システムディスクのIPL部分が破壊される現象をなくすようなっています。
6. モニタのXコマンドでaレジスタに【RETURN】だけ入力した場合でも内容がこわれないようになっています。
7. PC−8881とPC−8031−1Wが同時に接続されていても8インチのN88−DISK−BASICが正常に立ち上がるようになっています。
8. 【NEW ON 1】でPC−8031−2WのN−BASICのシステムディスクが正常に立ち上がるようになっています。
9. N−BASICモードにおいてウォームスタート（【STOP】＋【RESET】）をした後でもディスクをアクセスできるようになっています。
10. 漢字ROMがない場合でもPUT命令によりASCII文字（128種，4/1角）を表示できるようになっています。

ROM 1，2（N-BASICモニタ他）

Address	Ver1.0	Ver1.1
0084	3A	CD
0085	55	C1
0086	EB	7A
0B18	11	CD
0B19	00	BA
0B1A	C0	7A
2DC4	30	5E
2DC5	CC	06
2DC6	C7	5E
2DC7	26	06
2DC8	CD	F8
2DC9	12	10
2DCA	30	83
2DCB	2B	0F
2DCC	7E	41
2DCD	FE	7F
2DCE	30	EE
2DCF	28	0F
2DD0	FA	07
2DD1	FE	40
2DD2	2E	00
2DD3	C4	A0
2DD4	C7	18
2DD5	26	0C
2DD6	F1	0F
2DD7	28	03
2DD8	21	BC
2DD9	F5	03
2DDA	CD	18
2DDB	D8	01
2DDC	4D	F1
2DDD	3E	0B
2DDE	22	FF
2DDF	8F	DF
2DE0	77	A3
2DE1	23	A3
2DE2	F1	24
2DE3	36	5F
2DE4	2B	02
2DE5	F2	5F
2DE6	EC	A8
2DE7	2D	5D
2DE8	36	5E
2DE9	2D	5C
6003	22	C3
6004	F5	31
6005	F1	7A
629F	EE	20
62A0	6E	7A
64C2	D7	2C
64C3	6F	7A
6DC6	F1	5E
6DC7	FE	06
6DC8	07	F8
6DC9	30	10
6DCA	30	83
6DCB	D5	0F
6DCC	6F	41

Address	Ver1.0	Ver1.1
6DCD	26	7F
6DCE	00	EE
6DCF	29	0F
6DD0	29	07
6DD1	11	40
6DD2	C7	00
6DD3	6A	A0
6DD4	19	18
6DD5	D1	0C
6DD6	E5	0F
6DD7	23	03
6DD8	23	BC
6DD9	7E	03
6DDA	23	18
6DDB	66	01
6DDC	6F	F1
6DDD	C5	0B
6DDE	06	FF
6DDF	FF	DF
6DE0	7E	A3
6DE1	A7	A3
6DE2	CA	24
6DE3	CE	5F
6DE4	60	02
6DE5	CD	5F
6DE6	46	A8
6DE7	6E	5D
6DE8	04	5E
6DE9	38	5C
7A20	00	FE
7A21	00	0E
7A22	00	28
7A23	00	04
7A24	00	FE
7A25	00	09
7A26	00	20
7A27	00	01
7A28	00	5A
7A29	00	C3
7A2A	00	EE
7A2B	00	6E
7A2C	00	CD
7A2D	00	D7
7A2E	00	6F
7A2F	00	3F
7A30	00	C9
7A31	00	22
7A32	00	F5
7A33	00	F1
7A34	00	D9
7A35	00	E1
7A36	00	E5
7A37	00	11
7A38	00	7B
7A39	00	FF
7A3A	00	19
7A3B	00	7C
7A3C	00	B5

Address	Ver1.0	Ver1.1
7A3D	00	D9
7A3E	00	C2
7A3F	00	06
7A40	00	60
7A41	00	D5
7A42	00	11
7A44	00	F8
7A45	00	19
7A46	00	29
7A47	00	11
7A49	00	7B
7A4A	00	19
7A4B	00	D1
7A4C	00	7E
7A4D	00	23
7A4E	00	12
7A4F	00	13
7A50	00	7E
7A51	00	23
7A52	00	12
7A53	00	13
7A54	00	10
7A55	00	F6
7A56	00	F1
7A57	00	F1
7A58	00	E5
7A59	00	D5
7A5A	00	C5
7A5B	00	11
7A5C	00	F9
7A5D	00	E9
7A5E	00	21
7A5F	00	6C
7A60	00	7A
7A61	00	01
7A62	00	0F
7A64	00	ED
7A65	00	B0
7A66	00	C1
7A67	00	D1
7A68	00	E1
7A69	00	C3
7A6A	00	F9
7A6B	00	E9
7A6C	00	F3
7A6D	00	F5
7A6E	00	3A
7A6F	00	C2
7A70	00	E6
7A71	00	E6
7A72	00	FB
7A73	00	D3
7A74	00	31
7A75	00	32
7A76	00	C2
7A77	00	E6
7A78	00	F1
7A79	00	FB

Address				
7A7A	:	00	-	C9
7A7B	:	00	-	3E
7A7C	:	00	-	0B
7A7D	:	00	-	CD
7A7E	:	00	-	7C
7A7F	:	00	-	01
7A80	:	00	-	3E
7A81	:	00	-	07
7A82	:	00	-	CD
7A83	:	00	-	83
7A84	:	00	-	01
7A85	:	00	-	3E
7A86	:	00	-	EF
7A87	:	00	-	CD
7A88	:	00	-	83
7A89	:	00	-	01
7A8A	:	00	-	AF
7A8B	:	00	-	CD
7A8C	:	00	-	83
7A8D	:	00	-	01
7A8E	:	00	-	3E
7A8F	:	00	-	01
7A90	:	00	-	CD
7A91	:	00	-	83
7A92	:	00	-	01
7A93	:	00	-	CD
7A94	:	00	-	E9
7A95	:	00	-	01
7A96	:	00	-	2F
7A97	:	00	-	E6
7A98	:	00	-	F0
7A99	:	00	-	FE
7A9A	:	00	-	10
7A9B	:	00	-	C9
7A9C	:	00	-	3E
7A9D	:	00	-	0F
7A9E	:	00	-	18
7A9F	:	00	-	01
7AA0	:	00	-	AF
7AA1	:	00	-	F5
7AA2	:	00	-	3A
7AA3	:	00	-	C9
7AA4	:	00	-	ED
7AA5	:	00	-	FE
7AA6	:	00	-	FB
7AA7	:	00	-	28
7AA8	:	00	-	0F
7AA9	:	00	-	CD
7AAA	:	00	-	7B
7AAB	:	00	-	7A
7AAC	:	00	-	20
7AAD	:	00	-	0A
7AAE	:	00	-	3E
7AAF	:	00	-	17
7AB0	:	00	-	CD
7AB1	:	00	-	7C
7AB2	:	00	-	01
7AB3	:	00	-	F1
7AB4	:	00	-	CD
7AB5	:	00	-	83
7AB6	:	00	-	01
7AB7	:	00	-	F5
7AB8	:	00	-	F1
7AB9	:	00	-	C9
7ABA	:	00	-	CD
7ABB	:	00	-	A0
7ABC	:	00	-	7A
7ABD	:	00	-	11
7ABF	:	00	-	C0
7AC0	:	00	-	C9
7AC1	:	00	-	3A
7AC2	:	00	-	C7
7AC3	:	00	-	ED
7AC4	:	00	-	B7
7AC5	:	00	-	28
7AC6	:	00	-	11
7AC7	:	00	-	3E
7AC8	:	00	-	91
7AC9	:	00	-	CD
7ACA	:	00	-	29
7ACB	:	00	-	02
7ACC	:	00	-	3E
7ACD	:	00	-	04
7ACE	:	00	-	32
7ACF	:	00	-	CB
7AD0	:	00	-	ED
7AD1	:	00	-	AF
7AD2	:	00	-	CD
7AD3	:	00	-	7C
7AD4	:	00	-	01
7AD5	:	00	-	CD
7AD6	:	00	-	9C
7AD7	:	00	-	7A
7AD8	:	00	-	3A
7AD9	:	00	-	55
7ADA	:	00	-	EB
7ADB	:	00	-	C9
7ADC	:	00	-	CD
7ADD	:	00	-	A0
7ADE	:	00	-	7A
7ADF	:	00	-	C3
7AE0	:	00	-	82
7AE1	:	00	-	3C
7AE2	:	00	-	11
7AE4	:	00	-	C0
7AE5	:	00	-	21
7AE6	:	00	-	F0
7AE7	:	00	-	7A
7AE8	:	00	-	01
7AE9	:	00	-	06
7AEB	:	00	-	ED
7AEC	:	00	-	B0
7AED	:	00	-	C3
7AEF	:	00	-	C0
7AF0	:	00	-	AF
7AF1	:	00	-	D3
7AF2	:	00	-	31
7AF3	:	00	-	C3
7FF0	:	00	-	C3
7FF1	:	00	-	E2
7FF2	:	00	-	7A
7FF3	:	00	-	C3
7FF4	:	00	-	DC
7FF5	:	00	-	7A

＊7B00H〜7EFFHには、¼角文字のキャラクタデータが格納されている。(Ver1.1のみ)

ROM 3, 4 (N_{88}-BASIC)

Address		Ver1,0		Ver1,1
007D	:	00	-	3A
007E	:	00	-	E9
007F	:	00	-	E9
0080	:	00	-	FE
0081	:	00	-	02
0082	:	00	-	CC
0083	:	00	-	26
0084	:	00	-	E8
0085	:	00	-	7D
0086	:	00	-	C9
06B3	:	E1	-	C9
06B5	:	CD	-	06
06B6	:	D4	-	06
06B7	:	06	-	1A
06B8	:	20	-	4F
06B9	:	31	-	CD
06BA	:	D7	-	14
06BB	:	11	-	14
06BC	:	E5	-	B9
06BD	:	06	-	20
06BE	:	CD	-	2C
06BF	:	D4	-	23
06C0	:	06	-	13
06C1	:	3E	-	10
06C2	:	89	-	F4
06C3	:	CA	-	3E
06C4	:	CE	-	8D
06C5	:	06	-	C1
06C6	:	11	-	C3
06C7	:	E8	-	5E
06C8	:	06	-	07
06C9	:	CD	-	47
06CA	:	D4	-	4F
06CB	:	06	-	20
06CC	:	20	-	53
06CD	:	1D	-	55
06CE	:	3E	-	42
06CF	:	8D	-	F6
06D0	:	C1	-	AF
06D1	:	C3	-	2A
06D2	:	5E	-	16
06D3	:	07	-	EB
06D4	:	1A	-	22
06D5	:	B7	-	1B
06D6	:	C8	-	EB
06D7	:	4F	-	22
06D8	:	CD	-	1D
06D9	:	14	-	EB
06DA	:	14	-	22
06DB	:	B9	-	1F
06DC	:	C0	-	EB
06DD	:	23	-	CA
06DE	:	13	-	96
06DF	:	18	-	03
06E0	:	F3	-	C3
06E1	:	47	-	93
06E2	:	4F	-	03
06E3	:	20	-	2A
06E4	:	00	-	43
06E5	:	54	-	EC
06E6	:	4F	-	44
06E7	:	00	-	4D
06E8	:	55	-	C3
06E9	:	42	-	43
06EA	:	00	-	28
18B0	:	2A	-	32
18B1	:	58	-	FD
18B2	:	E6	-	EA
18B3	:	22	-	3A
18B4	:	1B	-	9F
18B5	:	EB	-	E6
18B6	:	21	-	B7
18B7	:	00	-	C2
18B8	:	80	-	06
18B9	:	5E	-	0B
18BA	:	23	-	C9
18BB	:	56	-	87
18BC	:	23	-	FE
18BD	:	23	-	FE
18BE	:	22	-	20
18BF	:	58	-	08
18C0	:	E6	-	3A
18C1	:	EB	-	7F
18C2	:	22	-	EF
18C3	:	54	-	E6
18C4	:	E6	-	20
18C5	:	F9	-	20
18C6	:	11	-	01
18C7	:	00	-	5F
18C8	:	FF	-	3A
18C9	:	19	-	53
18CA	:	22	-	F1
18CB	:	CC	-	C9
18CC	:	EA	-	7C
18CD	:	01	-	32
18CE	:	96	-	E9
18CF	:	09	-	E9
18D0	:	C5	-	C3
18D1	:	C3	-	90
18D2	:	21	-	72
18D3	:	4F	-	00
281C	:	7F	-	32
2849	:	3A	-	C3
284A	:	50	-	E3
284B	:	EC	-	06
310D	:	08	-	7F
3131	:	08	-	7F
3180	:	3A	-	CD
3181	:	53	-	BB
3182	:	F1	-	18
3706	:	04	-	05
39A6	:	B3	-	B2
39B2	:	B2	-	B0
45D1	:	37	-	3F
519D	:	32	-	CD
519E	:	FD	-	B0
519F	:	EA	-	18
53CF	:	93	-	CF
53D0	:	03	-	06
53DB	:	96	-	D0
53DC	:	03	-	06
7889	:	30	-	38
79D7	:	30	-	31

ROM 5 (N_{88}-BASIC)

Address		Ver1,0		Ver1,1
70F0	:	CA	-	28
70F1	:	F5	-	03
70F2	:	70	-	01
70F3	:	06	-	03
723E	:	7D	-	CD
723F	:	D3	-	7D
7240	:	E8	-	00
7241	:	7C	-	D3
7242	:	D3	-	E8
7243	:	E9	-	7C
7245	:	EA	-	E9
7246	:	00	-	D3
7247	:	00	-	EA
7264	:	90	-	CC
7265	:	72	-	18
7295	:	01	-	03
7296	:	28	-	38

付録16　N_{88}−DISK−BASIC（〔Feb 4, 1982〕 vs 〔Apr 24, 1982〕

	Feb.	Apr.		Feb.	Apr.		Feb.	Apr.
87D1 :	30	38	AD65 :	XX	5F	AD90 :	XX	CD
87F4 :	F5	CD	AD66 :	XX	EF	AD91 :	XX	D2
87F5 :	E5	80	AD67 :	XX	3A	AD92 :	XX	37
87F6 :	F1	54	AD68 :	XX	85	AD93 :	XX	CD
87F7 :	DA	2B	AD69 :	XX	EC	AD94 :	XX	47
87F8 :	01	D7	AD6A :	XX	CD	AD95 :	XX	38
87F9 :	05	B7	AD6B :	XX	CB	AD96 :	XX	E6
87FA :	21	C9	AD6C :	XX	3D	AD97 :	XX	40
8801 :	5D	00	AD6D :	XX	FE	AD98 :	XX	C2
8802 :	AD	AF	AD6E :	XX	02	AD99 :	XX	28
9AFF :	2A	CD	AD6F :	XX	30	AD9A :	XX	A7
9B00 :	86	5B	AD70 :	XX	13	AD9B :	XX	C1
9B01 :	EC	AD	AD71 :	XX	CD	AD9C :	XX	D1
A188 :	CD	3A	AD72 :	XX	67	AD9D :	XX	F1
A189 :	F3	85	AD73 :	XX	3D	AD9E :	XX	2A
A18A :	A1	EC	AD74 :	XX	3E	AD9F :	XX	86
A18C :	7E	CD	AD75 :	XX	04	ADA0 :	XX	EC
A18D :	CF	F3	AD76 :	XX	CD	ADA1 :	XX	C9
A18E :	29	A1	AD77 :	XX	94			
A18F :	F1	F5	AD78 :	XX	3C			
A190 :	E5	CF	AD79 :	XX	3A			
A191 :	6F	29	AD7A :	XX	5F			
A192 :	3A	D1	AD7B :	XX	EF			
A193 :	85	F1	AD7C :	XX	CD			
A194 :	EC	E5	AD7D :	XX	94			
A196 :	7D	7A	AD7E :	XX	3C			
A737 :	CD	00	AD7F :	XX	CD			
A738 :	EA	00	AD80 :	XX	7F			
A739 :	97	00	AD81 :	XX	3C			
AA44 :	80	F4	AD82 :	XX	18			
AA45 :	54	87	AD83 :	XX	12			
AA46 :	2B	28	AD84 :	XX	FE			
AA47 :	D7	DC	AD85 :	XX	03			
AD5B :	XX	F5	AD86 :	XX	20			
AD5C :	XX	D5	AD87 :	XX	13			
AD5D :	XX	C5	AD88 :	XX	3E			
AD5E :	XX	3A	AD89 :	XX	14			
AD5F :	XX	85	AD8A :	XX	CD			
AD60 :	XX	EC	AD8B :	XX	C9			
AD61 :	XX	CD	AD8C :	XX	37			
AD62 :	XX	D9	AD8D :	XX	3A			
AD63 :	XX	3D	AD8E :	XX	5F			
AD64 :	XX	32	AD8F :	XX	EF			

索　引

索　引

著者略歴

栗山　浩一　（くりやま　こういち）
1960年　福岡県生まれ
現　在　九州大学情報工学科在学
著　書　PC－Tech Know 8000 Vol.1（共著）

平松　達雄　（ひらまつ　たつお）
1961年　長崎県生まれ
現　在　九州大学医学部在学
著　書　PC－Tech Know 8000 Vol.1（共著）

松尾　篤弥　（まつお　とくや）
1960年　福岡県生まれ
現　在　九州大学情報工学科在学

本書の内容に関する御質問は，下記のシステムソフト福岡まで御願い致します．

〒810　福岡県福岡市中央区天神2丁目14－8

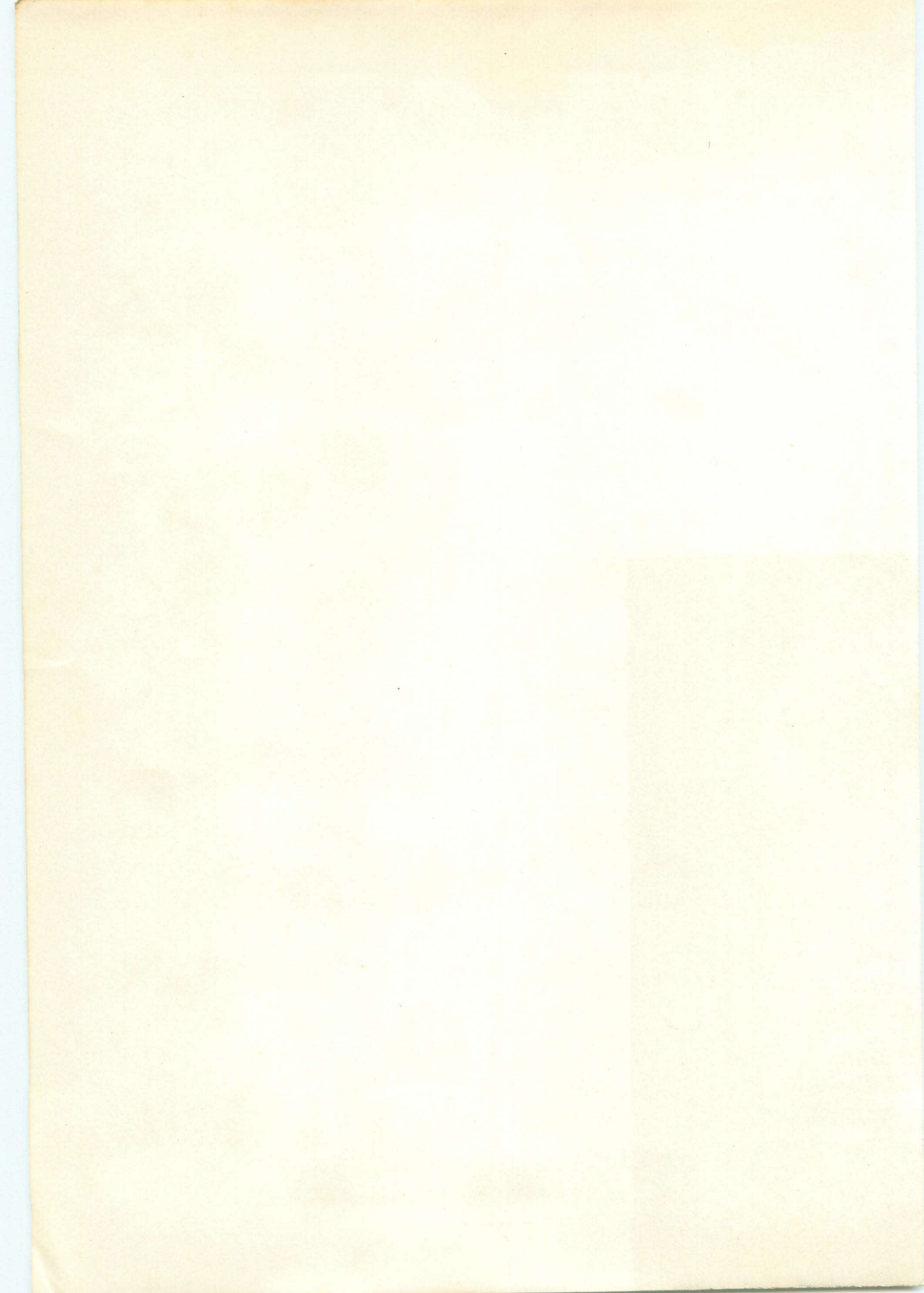
株式会社　システムソフト福岡

PCファミリー・テクニカル・ノウハウ集
PC-8800シリーズ編
PC-Techknow 8800 Vol.1

1982年12月20日　第1版第1刷発行
1984年12月5日　第1版第9刷発行
定価2,900円

共　著　栗山浩一・平松達雄・松尾篤弥
監　修　システムソフト
発行者　塚本慶一郎
発行所　株式会社アスキー
〒107　港区南青山5－11－5住友南青山ビル5F
振　替　東京7-57496
電　話　03－486－7111(代表)

印刷　凸版印刷

ISBN4-87148-292-8 C3055 ¥2900E

course of

The resul
Life of Ca
team ever
unchange
Observe
National

A Day in

監修
SYSTEM SOFT
発行
株式会社アスキー

定価2,900円

ISBN4-87148-292-8 C3055 ¥2900E